Aplicació d'un algoritme genètic per a l'optimització de les estacions de xarxes de bicis urbanes

Estudiant: **Marc Catllà Canónigo**
Grau en Enginyeria Informàtica
Intel·ligència artificial

Consultor: **Dr. David Isern Alarcón**
Professor responsable de l'assignatura: **Dr. Carles Ventura Royo**

Data Lliurament: 04/01/2022

FITXA DEL TREBALL FINAL

Títol del treball:	*Aplicació d'un algoritme genètic per a l'optimització de les estacions de xarxes de bicis urbanes*
Nom de l'autor:	*Marc Catllà Canónigo*
Nom del consultor/a:	*Dr. David Isern Alarcón*
Nom del PRA:	*Dr. Carles Ventura Royo*
Data de lliurament (mm/aaaa):	*01/2022*
Titulació o programa:	*Grau en Enginyeria Informàtica*
Àrea del Treball Final:	*Intel·ligència artificial*
Idioma del treball:	Català
Paraules clau	*genetic_algorithm, optimization, bike-sharing*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

Les xarxes de bici urbanes són un punt central en moltes ciutats, grans o petites, per tal de reduir la contaminació ja que permeten fer recorreguts curts i mitjans de forma còmode sense necessitat del vehicle privat o el transport públic. És per aquest motiu que esdevé molt important tenir una bona xarxa que permeti que els usuaris puguin realitzar els seus desplaçament i en facin ús.

En aquest treball s'analitza una xarxa ja existent i es desenvolupa i s'aplica un algoritme genètic que permet optimitzar les diferents estacions o punts de servei. Es creuen dades reals de les estacions d'una ciutat amb els viatges que s'hi van realitzar en un període de temps. La funció objectiu que es minimitza són els viatges fallits, que poden ser viatges que no s'haurien pogut realitzar perquè en aquell moment no hi havia bicis a l'estació o els viatges que no s'haurien pogut finalitzar perquè no hi havia espais lliures. Els cromosomes de la població tenen dos al·lels, un amb els espais de l'estació i l'altre amb el nombre de bicis a l'iniciar el dia.

Es tracta d'un problema complex de resoldre els resultats del qual han estat molt bons, ja que, tot i executar-se amb poques generacions i una població inicial no gaire gran, s'ha millorat molt el resultat si es fa servir la xarxa real. Per aquest motiu es podria considerar que es tracta d'un bon algoritme per resoldre aquest tipus de problemes d'optimització.

Abstract (in English, 250 words or less):

Urban bike-sharing is a central point in many cities in order to reduce pollution because it allows citizens to travel medium and short distances without using private car or public transport. For that reason, is important to have a good network to carry user's travels to make it useful.

This work analyses an existent network and develops and applies a genetic algorithm to optimize all the bike stations. Real data of the trips and the stations are joined in a period to evaluate the algorithm. The Objective function is to minimize failed trips, which means that a user could not start because there is not a bike at that station or that a user could not finish because there are not empty docks. Chromosomes have two alleles, one with docks at station and the other with the number of bikes at the beginning of the day.

The results of the algorithm applied to these complex problems are quite good because even executing with a few generations and small initial population it improves the real network results. Therefore, we could consider that a genetic algorithm is a good choice to resolve that kind of optimization problems.

Índex

1. Introducció 1
1.1 Context i justificació del Treball 1
1.2 Objectius del Treball 2
1.3 Enfocament i mètode seguit 2
1.4 Planificació del Treball 2
1.5 Breu sumari de productes obtinguts 4
1.6 Breu descripció dels altres capítols de la memòria 4
2. Teoria dels problemes d'optimització i els algorismes genètics 6
2.1 Els problemes d'optimització 6
2.2 Algorismes genètics 10
3. Estat de l'art 17
3.1 Taula resum 20
4. Disseny i implementació de l'algorisme 22
4.1 Anàlisis del dataset 22
4.2 Definició de l'algorisme genètic aplicat 32
4.3 Tractament de les dades 35
4.4 Creació del model d'optimització 37
4.5 Avaluació del model 43
5. Conclusions 51
6. Glossari 53
7. Bibliografia 54
8. Annexos 56

Llista de figures

Figura 1: Diagrama de Gantt amb la planificació de tasques 4
Figura 2: Exemple de resolució 9
Figura 3: Font:https://www.geeksforgeeks.org/np-completeness-set-1/ 11
Figura 4: Exemple de població en un AG 12
Figura 5: Etapes d'un AG 12
Figura 6: Font https://www.cienciadedatos.net/documentos/48_optimizacion_con_algoritmo_genetico 13
Figura 7: Font:https://www.cienciadedatos.net/documentos/48_optimizacion_con_algoritmo_genetico 14
Figura 8: Font: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3 15
Figura 9: Font: https://www.cienciadedatos.net/documentos/48_optimizacion_con_algoritmo_genetico 16
Figura 10: Exemple d'individu amb dos cromosomes 18
Figura 11: Crossover de dos punts 18
Figura 12: Cromosomes i operadors genètics 20
Figura 13: Resum del camp install_date, font:https://www.kaggle.com/pronto/cycle-share-dataset/version/2 23
Figura 14: Resum del camp current_dockcount, font:https://www.kaggle.com/pronto/cycle-share-dataset/version/2 24
Figura 15: Estacions de bici situades per longitut i latitut 24
Figura 16: Estacions de bici cancel·lades 25
Figura 17: Mapa de la zona d'estudi 25
Figura 18: Resum d'usertype, font:https://www.kaggle.com/pronto/cycle-share-dataset/version/2 4 27
Figura 19: Estacions amb més viatges finalitzats 27
Figura 20: Tipus d'usuaris de les estacions amb més viatges iniciats 28
Figura 21: Duració dels viatges en segons 28
Figura 22: Duració dels viatges sense ouliers 29
Figura 23: Viatges per hora del dia 29
Figura 24: Resum del camp starttime, font:https://www.kaggle.com/pronto/cycle-share-dataset/version/2 29
Figura 25Màxim Dades de temperatura, font: https://www.kaggle.com/pronto/cycle-share-dataset/version/2 30
Figura 26: Mitjana de temperatures, font: https://www.kaggle.com/pronto/cycle-share-dataset/version/2 31
Figura 27: Mínim de temperatura, font:https://www.kaggle.com/pronto/cycle-share-dataset/version/2 31
Figura 28: Mitjana d'humitat, font:https://www.kaggle.com/pronto/cycle-share-dataset/version/2 31
Figura 29: Població de cromosomes per l'AG 33
Figura 30: Exemple d'encreuament entre dos cromosomes 34
Figura 31: Exemple de mutació de dos cromosomes 34
Figura 32: Comparativa de les funcions de selecció rank i trunk 40

Figura 33: Evolució de la funció objectiu de l'AG en dos execucions 43
Figura 34: Evolució de la funció objectiu del millor cromosoma de cada generació..... 44
Figura 35: Resultats de cada estació per l'execució de 50 generacions 45
Figura 37: Total de viatges iniciats per estació..... 46
Figura 36: Total de viatges finalitzats per estació..... 46
Figura 38: Funció objectiu de viatges iniciats per estació 46
Figura 39: Funció objectiu de viatges finalitzats per estació 46
Figura 40: Execució de 26 generacions canviant el mes 47
Figura 41: Funció objectiu de viatges iniciats per estació 48
Figura 42: Funció objectiu de viatges finalitzats per estació 48
Figura 43: Evolució de les funcions objectius per mes..... 49
Figura 44: Evolució de les funcions objectius amb el total de viatges per mes 49
Figura 45: Balanç de la funció objectiu de viatges iniciats per estacions 50
Figura 46:: Balanç de la funció objectiu de viatges finalitzats per estacions 50

Llista de taules

Taula 1: Planificació de les tasques 3
Taula 2: Riscos en la planificació 4
Taula 3: Resum de les principals característiques de l'estat de l'art 21
Taula 4: Dataset de les estacions de bici 22
Taula 5: Dataset trips 26
Taula 6: Dataset de temps 30

1. Introducció

1.1 Context i justificació del Treball

La mobilitat a les ciutats, la forma com ens movem, és un factor molt important per tal de reduir la contaminació i l'emissió dels gasos d'efecte hivernacle. Una bona xarxa de transport públic permet reduir l'ús del vehicle privat i agrupar en un sol vehicle persones que realitzen un mateix desplaçament. Tot i així, el transport públic també genera contaminació, pel que per realitzar desplaçaments curts o de mitjana distància és interessant l'ús de les bicicletes. Per aquest motiu a la majoria de ciutats s'estan impulsant des de fa anys el sistemes de 'bicycle-sharing', on hi ha estacions de bicis repartides de forma estratègica dintre la ciutat per tal que els usuaris puguin agafar una bici d'una estació i deixar-la en una altre cobrint així la seva necessitat de desplaçar-se per la ciutat.
Les estacions de bicis és important que estiguin ben localitzades i que disposin del número de llocs i bicis necessaris per cobrir les necessitats del usuaris per tal que en facin ús. És a dir, si un usuari no té cap punt proper de bicis, o, si els punts propers estan buits, o, si els punts on vols deixar la bici estan plens, no li serà un servei útil i no el farà servir. En aquest treball no es realitzarà l'estudi d'on s'han de col·locar les estacions de bici o si les que hi ha estan ben situades, però sí que s'intentarà abordar la problemàtica de, donada una xarxa d'estacions de bici i les dades d'ús dels seus usuaris, optimitzar el nombre de bicis i de forats disponibles que haurien de tenir les estacions per tal de cobrir les necessitats dels usuaris. No és un problema trivial i hi ha multitud de literatura al respecte on els autors utilitzen diferents tipus d'algorismes per tal de resoldre'l.
Alguns autors utilitzen un algoritme de classificació per agrupar estacions amb un comportament similar (KNN) i després apliquen alguna heurística per optimitzar el nombre de bicis per estació [10]. En principi en aquest TFG no s'abordarà el fet de com redistribuir les bicicletes entre estacions com en alguns estudis s'ha realitzat [12]. La idea del treball és realitzar un estudi similar al que es va realitzar a la ciutat de Nova York [7] però aplicant un altre mètode d'optimització, en aquest cas un algorisme genètic, per calcular les bicis i els forats disponibles que hauria de tenir cada estació a l'inici del dia. Un cop implementat l'algorisme (si hi hagués temps) es podria fer com a l'estudi anterior i mirar les diferències entre les hores puntes amb la resta del dia. Hi ha força autors que han utilitzat els algorismes genètics per a resoldre problemes d'optimització relacionats amb la mobilitat, tal i com es mostra a la taula 1 d'un estudi del problema de ubicació i assignació en un sistema de bike-sharing [1], però la majoria són per resoldre el problema de minimitzar el temps total de ruta o el cost total del sistema. Tot i que ja hi hagi literatura al respecte d'utilització d'algorismes genètics per resoldre el problema que es planteja en el TFG [8], trobo que és igualment interessant realitzar l'estudi i veure el resultat que puc obtenir. D'aquesta forma podré entendre millor aquests tipus d'algorisme d'optimització i la complexitat que tenen, així com veure el resultat obtingut en una situació d'interès actual i amb dades reals.
El resultat que s'esperaria del treball seria, donat un data set real on tinguéssim unes estacions definides, un número de bicicletes i els desplaçaments que han realitzat els usuaris entre estacions, aplicar un algorisme genètic que permeti

optimitzar el nombre de bicis que hi ha d'haver a cada estació, de forma que en la majoria d'escenaris possibles hi hagués el mínim nombre d'usuaris que no puguin agafar la bicicleta d'una estació perquè no hi ha cap. Com a segon objectiu igualment important seria el de minimitzar el nombre d'usuaris que no puguin deixar la bicicleta perquè la estació està plena.

1.2 Objectius del Treball

Primaris:

- Aplicar un algorisme genètic al problema d'optimització del nombre de bicis i llocs en unes estacions de 'bike-sharing'.
- Obtenir un set de dades per a poder aplicar l'algorisme
- Tractar les dades de forma adient a la resolució del problema
- Avaluació del resultat de l'algorisme genètic

Secundaris:

- Entendre els problemes d'optimització
- Entendre els algorismes genètics
- Plantejar de forma adequada l'algorisme genètic per a la resolució del problema
- Definir de forma correcte la funció objectiu del problema d'optimització

1.3 Enfocament i mètode seguit

Degut a que els problemes d'optimització són força desconeguts per mi, i encara més els algorismes genètics, el primer que caldria fer és realitzar una bona recerca bibliogràfica per tal d'entendre bé les bases d'aquests. Un cop entès millor la teoria que hi ha darrere, el següent pas seria realitzar una recerca de l'estat de l'art en el problema que s'ha triat resoldre per tal de veure que hi ha fet i utilitzar el màxim del que es pugui en aquest treball, ja que l'objectiu és aplicar un algorisme genètic per a resoldre el problema, per tant es pot aplicar alguna de les metodologies que ja hi ha implementades. Finalment, s'ha de buscar si hi ha dataset a la xarxa que permetin realitzar el treball.

Un cop realitzada aquesta primera part del treball que correspondria a la 1ª fase del TFG s'hauria de tractar les dades de forma realista i plantejar el problema de forma adequada per a poder-lo realitzar. Aquest tipus de problemes d'heurística generalment són molt complexos de resoldre, pel que s'hauria d'adaptar el dataset pel que fa al nombre d'estacions, bicis, usuaris i trajectes, per tal que pugui ser resolt.

1.4 Planificació del Treball

Per a la resolució del TFG es plantegen les següents tasques a resoldre per tal de complir els objectius definits del projecte en el temps que dura:

PAC	Tasca	Descripció	Inici	Final	Dies
	1	**PLA DE TREBALL**	**05/10/21**	**12/10/21**	**7**
PAC 1	1.1	Definició de l'escenari a treballar	05/10/21	07/10/21	2
	1.2	Context actual	05/10/21	10/10/21	5
	1.3	Definició d'objectius	10/10/21	12/10/21	2
	1.4	Planificació	11/10/21	12/10/21	1
	2	**DESENVOLUPAMENT DEL TREBALL (1a fase)**	**13/10/21**	**15/11/21**	**33**
PAC 2	2.1	Revisió bibliogràfica dels problemes d'optimització	20/10/21	25/10/21	5
	2.2	Revisió bibliogràfica dels algorismes genètics	25/10/21	31/10/21	6
	2.3	Estat de l'art	02/11/21	09/11/21	7
	2.4	Revisió dels datasets disponibles	10/11/21	11/11/21	2
	2.5	Descripció i anàlisi del dataset triat	11/11/21	15/11/21	4
	3	**DESENVOLUPAMENT DEL TREBALL (2a fase)**	**16/11/21**	**13/12/21**	**27**
PAC 3	3.1	Definició de l'algorisme genètic aplicat al problema d'optimització del TFG	16/11/21	24/11/21	8
	3.2	Tractament de les dades del dataset	24/11/21	29/11/21	5
	3.3	Creació del model d'optimització	27/11/21	13/12/21	16
	3.4	Avaluació del model	02/12/21	13/12/21	11
PAC 4	**4**	**REDACCIÓ DE LA MEMÒRIA**	**14/12/21**	**04/01/22**	**21**
	4.1	Redacció de la memòria	05/10/21	04/01/22	91

Taula 1: Planificació de les tasques

Tal i com es pot veure a les tasques representades en un diagrama de Gantt senzill, Figura 1, hi ha tasques que es poden realitzar de forma paral·lela, com és el cas de la definició de l'escenari a treballar o la definició del context actual. També hi ha tasques que es realitzen de forma recursiva, per exemple el tractament de les dades del dataset i la creació del model d'optimització, o la creació del model i l'avaluació d'aquest. Pot ser que mentre s'estigui creant el model es vegi que s'ha de fer algun tractament extra de les dades i s'hagi de tornar a la tasca anterior, o que un cop avaluat el model aquest s'hagi de modificar. Per aquests motiu hi ha tasques que apareixen solapades alguns dies en el diagrama. Finalment, la tasca de redacció de la memòria es transversal al llarg de tot el semestre, per tant, tot i que hi ha unes dates que es deixen per acabar-la, aquesta ha de durar des de el principi fins el final:

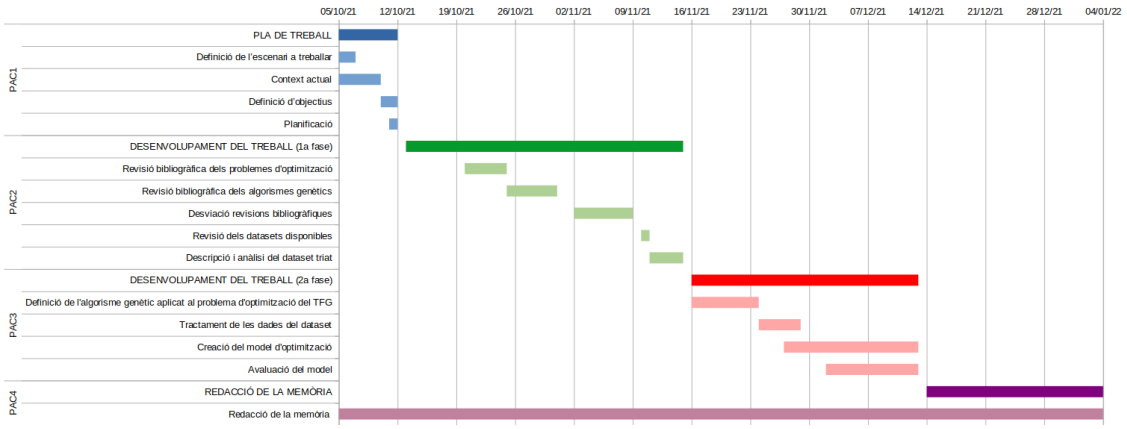

Figura 1: Diagrama de Gantt amb la planificació de tasques

1.4.1 Riscos

Finalment s'han identificat alguns dels riscos i dependències que hi ha en algunes de les tasques:

Tasca	Tipus de Risc	Descripció
2.4	Molt alt	Si no es troba un dataset no es pot implementar l'algorisme
2.6	Mig	Si no és un bon cas d'ús per aquest tipus d'algorisme serà complicat tractar les dades e implementar-lo
3	Molt alt	Dependència de les tasques 2.4 i 2.6
3.2	Baix	Durant els dies que dura la tasca hi ha el pont de desembre
3.3	Baix	Durant els dies que dura la tasca hi ha el pont de desembre
4	Mig	Durant els dies que dura la tasca hi ha dies festius importants i no es disposarà dels 21 dies que dura

Taula 2: Riscos en la planificació

1.5 Breu sumari de productes obtinguts

Al finalitzar el treball esperaria obtenir tres productes bàsicament:

- Una part més teòrica on es descrigui que són els problemes d'optimització i els algorismes genètics, així com la implementació d'aquest tipus d'algorismes al problema d'optimització de les xarxes de bicicletes.
- Un set de dades tractat per tal d'implementar un algorisme genètic al problema que es vol resoldre en aquest TFG.
- Un algorisme genètic avaluat amb el resultat sobre el problema d'optimització triat.

1.6 Breu descripció dels altres capítols de la memòria

- Desenvolupament del treball 1ª Fase

En aquest apartat hi anirà tota la part teòrica que hi ha darrere el TFG, així com l'anàlisi i descripció del data set triat per a realitzar l'algorisme genètic.

- Desenvolupament del treball 2ª Fase

La part pràctica del treball on es desenvoluparà l'algorisme i s'aplicarà a les dades tractades a l'apartat anterior. Trobarem un apartat de tractament de dades, un de desenvolupament de l'algorisme i un de avaluació d'aquest.

- Resultats i conclusions

En base al que s'hagi vist en els apartats anteriors es presentaran els resultats i les conclusions a les que s'hagi pogut arribar amb la realització d'aquest treball.

2. Teoria dels problemes d'optimització i els algorismes genètics

En aquest primer capítol s'abordarà de forma breu la part teòrica que hi ha darrere dels problemes d'optimització i dels algorismes genètics a mode d'introducció. L'objectiu és entendre que és la optimització, com es planteja un problema d'optimització, i com els algorismes genètics poden ser una bona eina per a resoldre'ls.

2.1 Els problemes d'optimització

L'optimització matemàtica es podria definir de forma genèrica com l'elecció de la millor solució, seguint uns criteris, entre varies solucions possibles aplicant models o equacions matemàtiques. Permet obtenir els factors principals d'un problema real, té aplicabilitat a multitud de diferents camps, i està format per uns objectius, unes variables i unes regles o normes que ha de complir la resolució. En la majoria de casos el que es tracta és de maximitzar o minimitzar el valor d'una funció triant els inputs adequats dintre del domini. Es podria formular de la següent forma [17]:

Donada una funció f: $A \rightarrow \mathbb{R}$

Trobar l'element $\mathbf{x}_0 \in A$ que compleixi una de les següents:

- $f(\mathbf{x}_0) \leq f(\mathbf{x})$ per tot $\mathbf{x} \in A$, *en el cas de la minimització*
- $f(\boldsymbol{x}_0) \geq f(\boldsymbol{x})$ *per tot* $\boldsymbol{x} \in A$, *en el cas de la maximització*

Generalment, s'ha establert el conveni de resoldre els problemes d'optimització com problemes de minimització, ja que a la pràctica, al ser la inversa l'una de l'altre es podria dir que és el mateix resoldre el màxim de la funció f que resoldre el mínim de la funció -f [4].

Definim les parts de la funció d'optimització [13]:

- f: se l'anomena funció objectiu, 'loss function', funció de cost (minimització) o 'fitness function' (maximització). És l'output que s'intentarà maximitzar o minimitzar, i per tant el model matemàtic que descriurà el comportament de la mesura d'efectivitat [15].
- x: inputs o variables que se li passen al model que poden ser variables continues o discretes.
- A: El domini de A se l'anomena espai de cerca, on trobem el que serien les solucions candidates.

- h(x) i g(x): no apareixen a la funció anterior ja que no són obligatòries que hi hagen aquestes clàusules. Són les regles que ha de complir la solució per a ser valida, on h són les regles d'igualtat i g de no-igualtat.

El primer pas per a resoldre un problema d'optimització és la formulació de la funció objectiu. Per a poder realitzar-ho s'ha de tenir un coneixement de les dades que es disposen i de quines variables o factors poden ser importants a la funció. El procés de selecció de variables és generalment un procés iteratiu en el que el resultat de la funció objectiu és el que marcarà si són bones variables per al model o no, per tant només es pot validar un cop realitzada la formulació i s'ha provat el model. Per tant, un cop és té el model s'ha de provar amb dades reals i fer un seguiment del seu comportament, per veure si realment és l'esperat en primer lloc, i quin dels model és el que millor s'ajusta a la realitat [15].

2.1.1 Tipus de problemes d'optimització

Generalment els problemes d'optimització es defineixen a partir d'una funció objectiu, però també trobem que hi poden haver problemes tinguin més d'una funció objectiu, en aquest cas s'anomenen problemes d'optimització multi-objectiu. En aquest tipus de problemes el que és òptim per una funció objectiu no ho és per una altre, pel que s'haurà de solucionar aquests conflictes entre funcions. La complexitat d'aquests tipus de problemes és major ja que s'ha de formular i trobar la que serà la millor solució global a base de donar prioritat a uns objectius per sobre d'uns altres i jugar amb les restriccions.

Una altre forma de classificació dels problemes d'optimització és tenint en compte si van succeint canvis en el temps. Trobem els estàtics, és a dir que o varien en el temps, i els dinàmics en el que s'haurà d'anar ajustant el model o la resolució a mesura que succeïxen els canvis[13].

Els sistemes poden ser determinístics si hi ha relacions de causa-efecte, o estocàstics si trobem que hi ha aleatorietat o probabilitat.

Finalment, trobem que els problemes es poden resoldre amb equacions lineals i els que es resolen amb equacions no-linears.

2.1.2 Resolució de problemes d'optimització

Hi ha molts subcamps i moltes formes diverses de resoldre els problemes d'optimització, un tipus d'agrupació que es pot fer es segons el nombre de passos que s'han de realitzar per resoldre'ls:

- Algorismes que es resolen amb un determinat nombre de passos.
- Algorismes iteratius que executen els mateixos passos diverses vegades.
- Algorismes heurístics que donen una solució aproximada al problema.

Alguns altres exemples d'agrupacions dels d'algorismes d'optimització són:

- Optimització lineal:

La funció objectiu i totes les variables i restriccions són lineals. Aquests problemes per tant es resolen amb equacions lineals amb igualtats i desigualtats de variables en primera potencia. Aquest tipus d'algoritmes també es podrien resoldre de forma gràfica representant les línies en els eixos x-y. Veiem un exemple extret d'una pàgina de la xarxa [15] on es vol calcular la producció de materials (X1 i X2) que tenen un benefici de 5 i 3 respectivament, per tant la funció objectiu serà:

maximitzar f: 5X1 + 3X2,

i tenim les següents restriccions:
2X1 + X2 £ 40
X1 + 2X2 £ 50
X1 i X2 £ 0

Si es resolen les equacions anteriors per els valors màxims possibles que pot tenir una variable tenint en compte que l'altre valgui 0, veiem que segons la restricció el valor de l'altre variable pot no ser factible per l'altre restricció. És a dir, si X1 val 0, a la primera restricció X2 pot valdre 40 i seria factible, però quan anéssim a mirar la següent restricció per X1=0 i X2=40 el resultat seria superior a 40 i per tant no seria factible. El mateix succeïx amb X2=0 i X1=50 que seria factible per la segona restricció i el resultat de la funció objectiu seria 250 però que no es factible per la primera restricció. Si tant X1 com X2 valen 0 el resultat final serà 0. Hi ha moltes possibilitats valides com X1 = 0 i X2 = 25 que dóna uns beneficis de 75, o X1=20 i X2=0 que dóna un benefici de 100, però en aquest problema, el resultat òptim surt de resoldre el sistema d'equacions:

$$\left.\begin{array}{l} 2X1 + X2 = 40 \rightarrow\ 2X1 + X2 = 40 \\ X1 + 2X2 = 50 \rightarrow -2X1 - 4X2 = -100 \end{array}\right\} -3X2 = -60 \rightarrow X2=20 \rightarrow X1=10$$

Aquest resultat ens dóna un benefici de 5X1 + 3X2 = 110, que seria l'òptim per aquest problema.

Si ho representem gràficament veiem el valor òptim és X1=10 i X2=20:

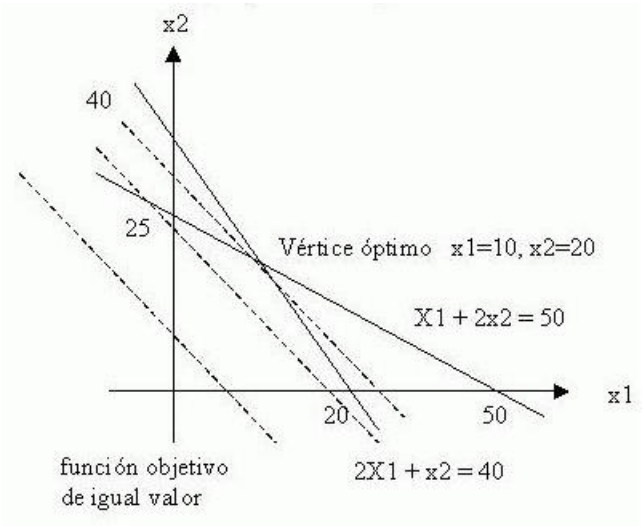

Figura 2: Exemple de resolució

En aquest cas que només tenim dues variables de decisió el punt on es creuen les dues rectes de les restriccions és l'òptim per a resoldre el problema.
Hi ha programes i softwares que amb una sintaxis molt similar a la exposada són capaços de resoldre els problemes lineals, com per exemple el programa LINDO (Linear Interactive Discrete Optimization), on passant-li la sentència
{MAX 5X1 + 3X2, S.T 2X1 + X2 < 40 X1 + 2X2 < 50, END}
resoldria el problema. En aquest cas era senzill, però en sistemes amb més variables i més restriccions aquest tipus de software pot ser de gran ajuda. Aquest tipus de software es pot utilitzar en algoritmes d'altre tipus.

- Optimització estocàstica:

Genera i utilitza variables aleatòries que apareixen a la formulació del problema per resoldre'l. Generalment l'aleatorietat apareix a la funció objectiu o les restriccions i, per tant, canviar els resultats amb cada resolució. Aquests mètodes de resolució es poden dividir entre problemes d'un sol escenari i problemes multi-escenari. Un exemple de com es formularia una funció objectiu per un problema un sol escenari seria el següent [6]:
Sent X el domini de totes les decisions factibles i x una decisió específica, i sent ξ la informació aleatòria del problema que només tenim disponible després de la decisió presa, trobar la solució que minimitzes la funció objectiu F. El resultat serà una funció que minimitzi el valor esperat (E):

$$\zeta^* = \min_{x \in \mathcal{X}} \{f(x) = \mathbb{E}[F(x, \xi)]\}$$

Aquest tipus d'optimització tenen un fort component matemàtic i estadístic i ha de portar implícites una sèrie d'assumpcions de com es genera o es processen les dades o de quin tipus es tracten. La optimització estocàstica multi escenari és més complexa i busca trobar una seqüencia de decisions que minimitzi la funció objectiu esperada. Es complex de formular ja que les decisions i els resultats aleatoris a cada temps afecta el valor de futures decisions. La formulació es similar a la d'un sol escenari però amb més estats i períodes (T), veiem un exemple de formulació estocàstica multi escenari [12]:

$$\zeta^* = \min_{x_0 \in \mathcal{X}_0} \mathbb{E}\left[\inf_{x_1 \in \mathcal{X}_1(x_0,\xi_1)} F_1(x_1,\xi_1) + \mathbb{E}\left[\cdots + \mathbb{E}\left[\inf_{x_T \in \mathcal{X}_T(x_{0:T-1},\xi_{1:T})} \gamma^{T-1} F_T(x_T,\xi_T)\right]\right]\right]$$

- Meta-heurística:

Els problemes d'optimització amb heurístiques o meta-heurístiques tracten de resoldre'ls amb la millor relació eficàcia-resultat. És a dir, es tracta de trobar una solució òptima (no cal que sigui la més òptima) en un temps o amb uns recursos que es considerin que són òptims també. Per tant, un cop s'hagi trobat una solució prou valida pel problema deixarà de seguir buscant. Els algorismes evolutius i més concretament els algorismes genètics entrarien dintre del grup de la optimització amb meta-heurístiques, es veuran en detall a l'apartat següent

2.2 Algorismes genètics

- Optimització robusta:

Similar a la programació estocàstica, tracta de trobar la millor solució d'un escenari canviant que pot tenir varies solucions òptimes en funció de l'escenari en que es trobi. Així, aquests tipus d'algorismes poden parar de buscar solucions abans de trobar la més òptima [9].

Hi ha més grups d'agrupació dels algorismes d'optimització, però aquí només s'han descrit alguns dels que s'ha trobat que poden resultar més interessants.

2.1.3 Camps d'aplicació

Els mètodes d'optimització es poden aplicar en camps molt diversos com la logística en el que es busca les millors rutes de repartiment, les finances on es busca maximitzar els beneficis o reduir els costos, fabricació de productes per calcular la producció per satisfer la demanda sense sobredimensionar l'estoc entre d'altres [5], en recursos humans on s'intenta cobrir les necessitats d'una empresa sense sobredimensionar la plantilla (per exemple instal·ladors de telefonia) [15].
Destacar un altre camp molt interessant on l'aplicació dels algorismes d'optimització pot ser de gran utilitat són a l'enginyeria elèctrica, ja que en el transport de l'electricitat des de el punt d'origen fins als punts finals es va perdent molta energia. Per aquest motiu pot ser molt útil els algorismes d'optimització que calculin el millor escenari possible abans de muntar tota la xarxa.

2.2 Algorismes genètics

Els algorismes genètics són un tipus d'algorismes evolutius, aquests, tal i com el seu nom indica, estan basats en la teoria de l'evolució biològica. En aquest primer apartat definirem que són i per a que s'usen els algorismes evolutius i després ens centrarem pròpiament en els algorismes genètics.

2.2.1 Introducció als algorismes evolutius

En l'àmbit de la intel·ligència artificial és habitual que hi hagi repliques del funcionament biològic, tal i com succeeix amb les xarxes neuronals. Els

algorismes evolutius han agafat el model de la teoria de Darwin de l'evolució on tenim una espècie, en aquest cas se l'anomena població, que va millorant els seus espècimens generació rere generació per tal de trobar una solució factible a un problema complex en un temps raonable.

Els algorismes evolutius s'usen en problemes que no es poden resoldre en un temps polinomial (problemes de de complexitat P). És a dir, per resoldre problemes en que no es pot definir el temps que es tardarà en resoldre's amb una funció polinomial a partir d'una variable que defineixi la mida del problema. Dintre dels problemes que no són de tipus P trobem els NP-complets i els NP-difícil. Els problemes NP-complets es poden verificar però no resoldre amb temps polinòmics i es poden reduir a problemes de tipus P. Els problemes NP-difícil en canvi no tots es poden verificar amb temps polinòmics però sí que també es poden descomposar en problemes de tipus P. Veiem una imatge que resumeix els 4 tipus de grups:

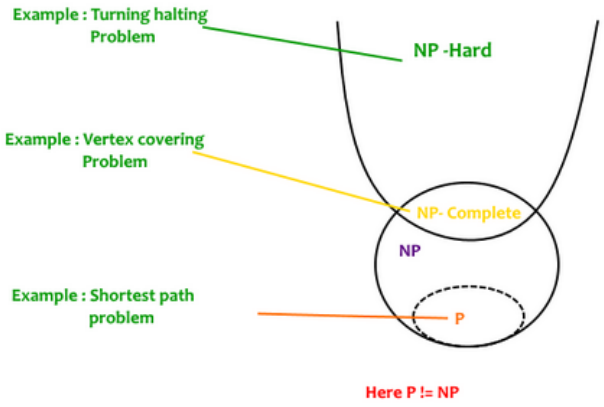

Figura 3: Font:https://www.geeksforgeeks.org/np-completeness-set-1/

Tal i com s'ha mencionat anteriorment, els algoritmes evolutius es basen en la teoria de la evolució, i tenen 4 fases que s'executen fins a trobar la millor solució possible. Els passos que segueixen són la iniciació, la selecció, els operadors genètics (algorismes genètics) o d'evolució (altres algorismes) i la finalització [14]. La funció comença per la primera fase, va iterant la segona i tercera, al llarg de la evolució, i es finalitza a la quarta fase en funció del nombre d'iteracions màxim definit o en el moment en que s'ha arribat a uns valors acceptables per la funció objectiu.

2.2.2 Els algorismes genètics

Un cop introduït els algorismes evolutius i les seves fases, podem veure que en el cas dels algorismes genètics les fases són les mencionades a l'apartat anterior amb alguna particularitat pel que fa a la població i en els operadors genètics. La

població en aquest tipus d'algorisme està formada per cromosomes, on cada cromosoma està compost per diferents gens.

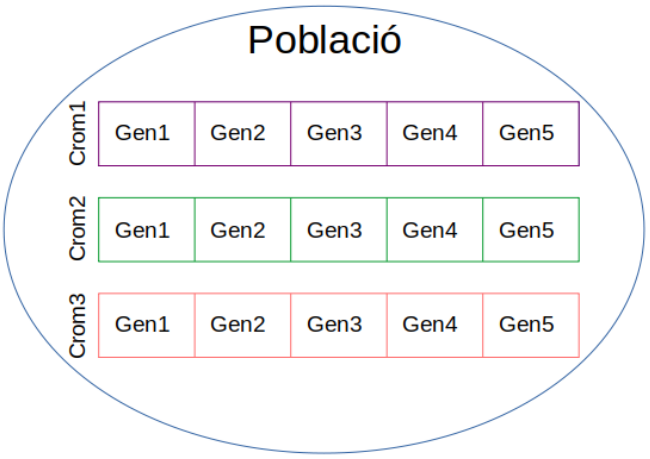

Figura 4: Exemple de població en un AG

Cada cromosoma és una possible solució al problema, i cada gen és un component de la funció objectiu, així els cromosomes que suposin una millor solució al problema són els que sobreviuran. Els millors individus de cada generació es reprodueixen entre ells generant nous individus amb característiques que seran la combinació dels progenitors més alguna mutació. Aquest pas es va repetint de nou cada generació transmitent així les seves característiques generació rere generació fins que trobem la millor solució.

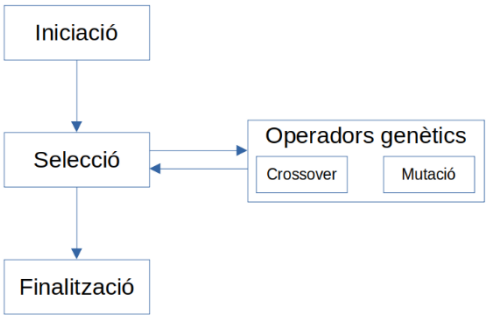

Figura 5: Etapes d'un AG

Descrivim amb més detall totes les fases:

- Iniciació o inicialització:

Es tracta de crear, ja sigui de forma aleatòria o no, una primera població que estarà formada per solucions possibles per resoldre el problema, on cada solució per tant serà un membre de la població. De cara a trobar la millor solució possible al llarg del procés evolutiu, és important que la variabilitat de solucions sigui gran

ja que a base de seleccionar els millors membres de la població es vol arribar a una població que sigui òptima per a la resolució del problema.

- Selecció

Es calcula quins són els millors membres de la població a partir d'una funció 'fitness' creada per resoldre el problema en qüestió. És important que sigui una funció representativa de les dades i ens dirà de forma numèrica si un membre està més a prop o més lluny de la solució respecte la resta. Es calcula el valor de fitness de cada membre i es queda amb un top membres, és a dir passen a la següent generació els millors individus. Com en qualsevol problema d'optimització es buscarà minimitzar o maximitzar una funció, per tant si el que es vol es minimitzar els millors individus seran els que tinguin un fitness més baix i si es vol maximitzar seran els que tinguin un valor més alt. Hi ha diferents mètodes per fer la selecció [2]:

- Truncada: Es descarten els individus amb pitjor fitness i amb la resta es van realitzant seleccions aleatòries.
- Rank: S'ordenen els individus per valor de fitness i la probabilitat de selecció serà major com millor fitness tingui.
- Competitiva (tournament): es seleccionen aleatòriament dues parelles d'individus i de cada parella es selecciona el que tingui millor fitness. El que té millor fitness entre els guanyadors és el que se selecciona.
- Ruleta: La probabilitat de selecció d'un individu és proporcional al fitness que tingui dividit entre el fitness total de tots els individus.

Cada mètode té els seus avantatges i inconvenients en funció del tipus de població que hi hagi. Per exemple, si no hi ha molta diferència de fitness entre individus, amb el mètode rank és més probable que es seleccionin més sovint els individus amb millor fitness i la següent generació estigui formada per uns pocs progenitors:

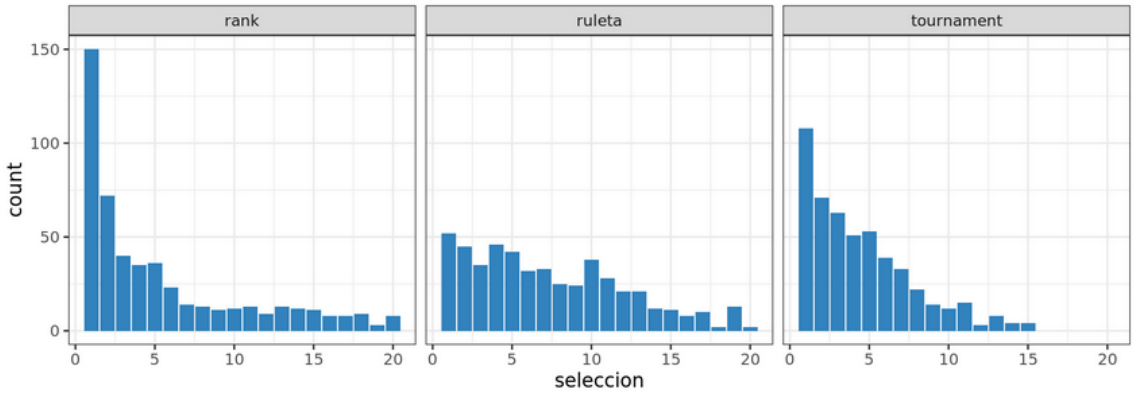

Figura 6: Font https://www.cienciadedatos.net/documentos/48_optimizacion_con_algoritmo_genetico

Mentre que si hi ha molta diferència entre el fitness dels diferents individus amb el mètodes de ruleta la probabilitat dels que tenen un millor fitness serà molt més elevada (la probabilitat serà el fitness de l'individu entre el total), i la següent generació estarà formada amb uns pocs progenitors. És important doncs conèixer la població d'individus i utilitzar el mètode més adequat per fer la selecció. Un cop s'han seleccionat els millors individus, aquests es creuen per obtenir una nova generació en la següent fase de l'algorisme, o es finalitza amb el resultat obtingut.

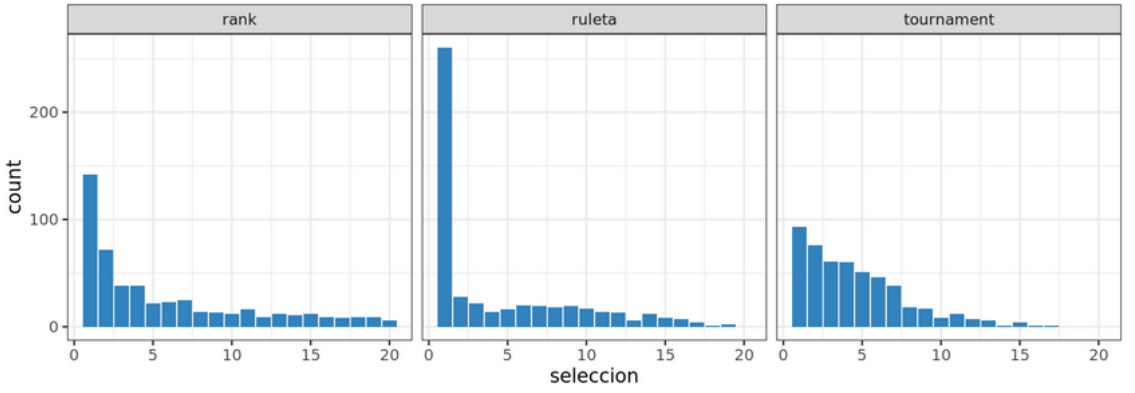

Figura 7:
Font:https://www.cienciadedatos.net/documentos/48_optimizacion_con_algoritmo_genetico

- Operadors genètics

En aquest encreuament hi ha una combinació dels gens dels individus, que són les característiques o qualitats de cada parent. A més de la combinació de de qualitats del parents ('crossover') hi ha una part de mutació en que s'introdueix nou 'material genètic' a la població de forma probabilística. Hi ha diferents estratègies per realitzar el crossover entre individus [2]:

- Crossover uniforme: A partir de dos parents es genera un únic individu que agafa cada característica (gen) d'un dels dos parents. Aquesta selecció del gen es fa de forma probabilística i pot ser que sigui la mateixa probabilitat per els dos parents o que sigui en funció del fitness de cadascun.
- Crossover a partir d'un punt. Es selecciona una posició dintre del cromosoma a partir de la qual s'intercanviaran tots els gens entre els dos individus. Es generen per tant dos nous individus:

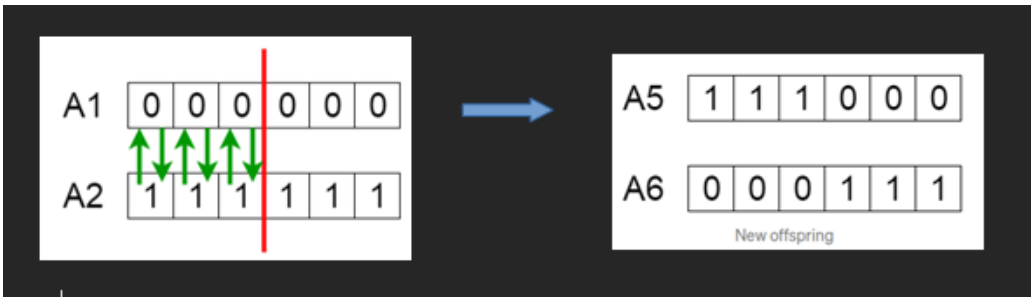

Figura 8: Font: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

- Creuament a partir de varis punts: Igual que el creuament anterior però en aquest cas es seleccionen diferents punts dintre el cromosoma. També es generen dos nous individus amb aquest sistema.

És habitual optar per una únic estratègia de crossover, però també es podria combinar-les aleatòriament a l'algorisme. Un cop tenim generats els nous individus, aquests poden 'patir' una mutació als gens per tal que hi hagi variabilitat i l'algorisme no caigui en mínims locals en que tots els individus són molt similars entre generacions. Aquesta mutació es produeix a cada individu amb una probabilitat que s'assigni i tal i com succeeix amb el crossover hi ha diferents mètodes de mutació [2]:

- Distribució uniforme: la mutació del gen que correspongui s'aconsegueix sumant un valor extret d'una distribució uniforme, per exemple [-1,1].
- Distribució normal: la mutació del gen es fa sumant un valor extret d'una distribució normal. La distribució acostuma a estar centrada a 0, pel que habitualment es sumarà o restarà un valor baix pròxim a 0, però que depenent de la desviació estàndard que tingui es podran produir grans variacions del gen.
- Aleatori: la mutació del gen es fa amb un valor aleatori dintre un rang de valors.

Després que han actuat els operadors genètics es torna al procés de selecció per escollir els que seran la nova generació o per finalitzar el procés si ja hi ha individus que tinguin un bon resultat.

- Finalització

Finalment, si el resultat entra dins de l'ombral establert o si s'ha arribat a un màxim d'iteracions establert, l'algorisme arriba al final i el resultat es trobarà dins de la última població obtinguda. Durant l'execució de l'algorisme es pot

emmagatzemar el valor del fitness global de tots els individus i veure gràficament com aquest ha anat millorant amb cada generació:

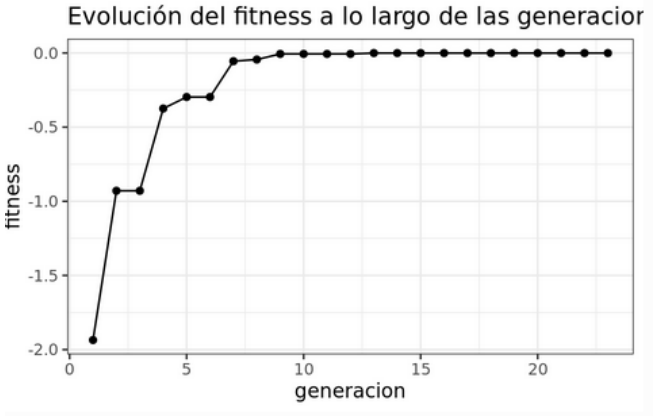

Figura 9: Font: https://www.cienciadedatos.net/documentos/48_optimizacion_con_algoritmo_genetico

Tal i com veiem a la imatge anterior, amb el pas de les generacions el fitness cada cop és més baix, pel que vol dir que potser no hem arribat al resultat òptim però estem molt a prop. Els algorismes genètics poden tenir un cost computacional molt alt ja que han d'avaluar la funció objectiu moltes vegades i aplicar la selecció i els operadors genètics. Degut a aquest alt cost, s'acostuma a posar el que s'anomena una parada temprada, és a dir, un ombral a partir del qual si després de x generacions no s'està generant un canvi mínim s'atura l'execució.

3. Estat de l'art

En aquest apartat es revisaran els treballs relacionats més rellevants que poden servir de base per a resoldre el problema d'optimització del nombre de bicis i llocs lliures en unes estacions de 'bicycle-sharing'.

Simulation optimization for a large-scale bike-sharing system (Jian et al., 2016) [8].

L'estudi és molt similar al que es vol realitzar en aquest TFG, on, per la ciutat de Nova York es vol optimitzar tant els llocs de cada estació com el nombre de bicis a l'inici del dia per tal que es minimitzin el nombre d'usuaris que es queden sense. En una ciutat com NY per la nit els és més fàcil moure's en camió per redistribuir les bicis entre emplaçaments, pel que és un problema molt útil de resoldre. Degut a la gran quantitat d'estacions de bicis que hi ha, 466, i el nombre de bicis, 6.074, és un problema que no es pot resoldre amb mètodes estocàstics com el gradient-search o el random-search. Per aquest motiu s'usarà un algorisme de cerca heurística que anirà actualitzant el nombre de bicis i espais entre estacions. Per l'estudi s'utilitzen dades reals dels viatges que es realitzen durant 14 dies laborables a l'any 2015.

L'article comença fent unes aproximacions primer situant un nombre de bicis proporcional a cada punt. Posteriorment calcula un model com si no hi hagués limitacions de espais o de bicis i valida que el número de bicis i espais que necessitaria per cobrir tota la demanda és molt més que els que realment hi ha. La funció objectiu que s'intenta minimitzar són el nombre d'usuaris que no poden agafar una bici perquè no hi ha, el nombre d'usuaris que no poden deixar la bici perquè no hi ha lloc i el nombre d'usuaris que no troben bici en 3 punts diferents:

$$\min_x f(x) = E_x[\text{\#failed-starts}] + E_x[\text{\#failed-ends}] + E_x[\text{\#bad-ends}]$$

A partir de la funció objectiu es realitzen diverses cerques heurístiques tenint en compte diferents períodes de temps. Per exemple tenint en compte només la hora punta del matí, 6-10 am, o tenint en compte el dia sencer. A cada iteració es troba una solució i s'avalua, si millora la funció objectiu es queda com a solució. L'avaluació del model es realitza amb un model de simulació a partir de les dades dels viatges reals entre estacions que calcula els potencials ciclistes a les estacions. L'algorisme aplicat estableix una parada si passades 200 iteracions no s'ha millorat. Els resultats del model només aplicat a les hores puntes del matí obté millors resultats que quan s'aplic a atot el dia ja que el comportament canvia molt per a estacions del matí a a la tarda. Per solucionar aquesta limitació s'hauria d'aplicar un algorisme de recol·locació de les bicis.

Hybrid Algorithm for Route Design on Bus Rapid Transit Systems (Walteros et al., 2013) [16].

S'utilitza un algorisme genètic com a part d'un algoritme que permeti optimitzar les rutes de bus ràpid per tal que es minimitzi el temps de viatge dels passatgers i de l'autobús. Haurà de satisfer com a regles o 'constraints' que sigui capaç de cobrir totes les rutes, sigui capaç de cobrir tota la demanda de passatgers i les freqüències de busos.

Es tracta d'un problema molt més complex que el que es pretén abordar en aquest treball, però és molt útil per entendre com planteja l'algoritme genètic. Trobem que hi ha vaires rutes de bus a cobrir, i que cada individu té dos cromosoma amb la ruta que cobreix. Els gens de cada cromosoma representen una parada i tenen un número binari que indica si el bus s'atura a la parada o no, veiem la imatge extreta de l'article:

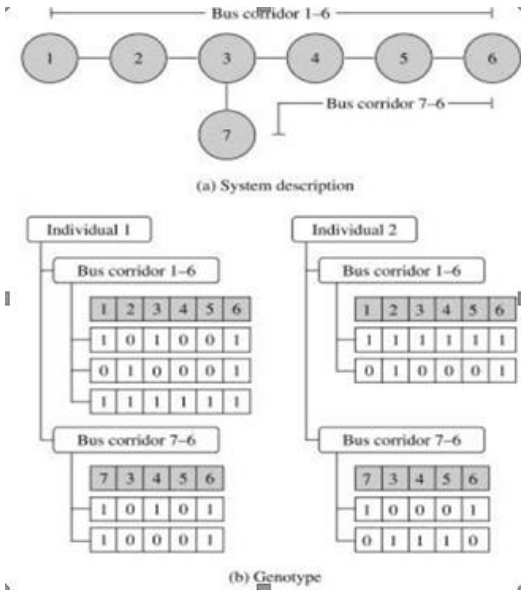

Figura 10: Exemple d'individu amb dos cromosomes

El crossover s'ha realitzat a partir de 2 punts, generant dos nous individus de cada dos parents:

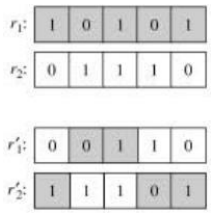

Figura 11: Crossover de dos punts

En aquest article, l'algorisme genètic l'utilitzen per donada una població de rutes possibles que han calculat amb altres algoritmes, trobar les més òptimes. Els resultats els han validat amb valors reals resolent una optimització linear. Els resultats que van obtenir considero que són molt bons, ja que de 14 problemes, 4 es van resoldre de la forma més òptima i 8 de forma probablement òptima amb una confiança del 0,2%.

Tesis Doctoral: Modelos de optimización para planificación y gestión opeartiva de sistemas de bicicleta pública (Juan Pablo Romero Junquera, 2013), pàgines 144-150 [11].

A l'apartat 4.4 de la tesi, pàgina 103, s'aborda la problemàtica de la optimització de de la gestió de les bicicletes en una xarxa pública. Els llocs que han d'haver ocupats i disponibles al principi del dia, el balanç que es va fent dels punts i com s'hauria de calcular per redistribuir el nombre de bicis mitjançant furgonetes al llarg del dia. Com realitzar la redistribució i el càlcul de millors rutes que realitza l'estudi no entrarien en l'àmbit d'aquest treball final de grau.

A l'apartat 5.4, primer calcula el balanç a cada punt sense fer redistribució i es busca el seu màxim negatiu i es suma a tots els del balanç per trobar la capacitat del punt de préstec que hauria de ser el màxim dels nous valors obtinguts. Amb aquest senzill càlcul per a cada punt troba el nombre total d'anclatges que hauria de tenir i el nombre de bicis a l'inici del dia. Quan es mira per a tots els punts, es troba que el model queda sobredimensionat amb masses bicicletes i punts d'anclatge. Decideix posar restriccions del nombre d'anclatges per punt i tornar a calcular els valors per començar amb la part de redistribució.

En aquesta tesi no es fa servir un algorisme genètic però pot ser molt útil per plantejar la funció objectiu i les restriccions del model.

A new genetic approach for transport network design and optimization (S.Dinu i G.Bordea, 2011) [3].

Proposa un algorisme genètic diploide, similar al que succeeix biològicament amb dos cromosomes per individu enlloc de un (haploide), per resoldre l'optimització d'una xarxa de transport. Presenta un algorisme que modifica el crossover afegint la meiosi, procés de divisió cel·lular de les cèl·lules sexuals, i una modificació en el procés de mutació. En el procés de meiosi cada cromosoma passa per dos meiosis generant 4 cromàtides, per tant tindrem 8 cromàtides per individu, les cromàtides homòlogues entre individus s'aparellen formant un cromosoma. El resultat és un model meta-heurístic que millora la cerca global a l'inici i la cerca local a les últimes generacions respecte un algorisme genètic tradicional.

Spatial Cluster-Based Model for Rebalancing Bike Sharing Problem (Lahoorpoor et al., 2019) [5]

En aquest article s'aborda la redistribució estàtica en un sistema de bike-sharing fent servir algorismes de clustering per després fer el rebalanceig de les bicis amb un algorisme genètic. És una aproximació que pot ser molt útil en aquests TFG per incloure com a fase prèvia a l'algorisme genètic, per exemple.

Amb els punts de bicis en clústers, l'article planteja el rebalanceig intra-clústers i inter-clústers, i l'optimitza amb un algorisme genètic, obtenint bons resultats pel que fa a la distància recorreguda. Veiem una imatge de l'article amb un exemple de cromosomes i els operadors genètics aplicats:

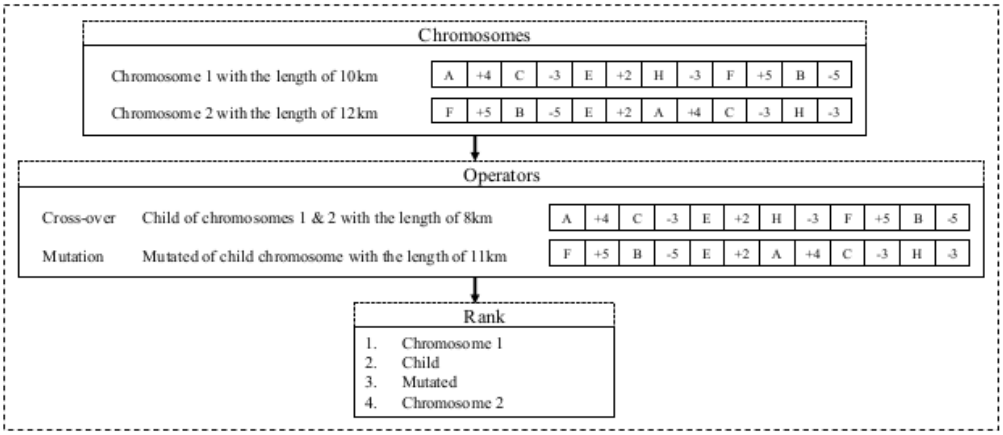

Figura 12: Cromosomes i operadors genètics

3.1 Taula resum

En aquest sub-apartat exposarem una taula resum de les principals característiques dels treballs revisats a l'estat de l'art que s'han tingut en compte a l'hora de realitzar el TFG:

Autors	Característiques
Jian et al., (2016) [8]	• Optimització les estacions de bicis de NY, número de llocs i bicis, per minimitzar els usuaris que no poden fer ús. • Elevat nombre d'estacions i bicis. • Utilització de dades de viatges reals per l'estudi. • Funció objectiu molt senzilla basada en el nombre de viatges que no es poden iniciar o finalitzar. • Resultats diferents aplicant el model a diferents horaris del dia.
Walteros et al., (2013) [16]	• Optimització de rutes de bus ràpid. • Algoritme complex amb multitud de 'constraints'. • Cromosomes dobles per individu. • Cada posició d'un gen és una parada de bus concreta

	• El model s'aplica a una població de rutes possibles. • Validació amb dades reals de viatges.
Juan Pablo Romero Junquera, (2013) [11]	• Bona introducció teòrica al problema d'optimització de la gestió de bicicletes d'una xarxa pública. • Estudi de l'òptim sense limitacions mostra que el problema es sobredimensiona. • Plantejament de funcions objectius i restriccions d'un model d'optimització.
S.Dinu i G.Bordea, (2011) [3]	• Model de cromosomes diploides (dos al·lels per individu). • Modificació del procés de 'crossover', afegeix la meiosi i modifica el procés de mutació. • L'algorisme genètic replica de forma més similar a la divisió cel·lular sexual obtenint bons resultats en la cerca global inicial i en les últimes generacions.
Lahoorpoor et al. , (2019) [5]	• Optimització d'un sistema de 'bike-sharing'. • Clúster previ de les estacions. • Re-balanceig de les estacions inter-clústers i intra-clústers.

Taula 3: Resum de les principals característiques de l'estat de l'art

4. Disseny i implementació de l'algorisme

En aquests apartat es desenvoluparà l'algorisme genètic que permeti resoldre el problema d'optimització del nombre de bicis i espais buits a les estacions de 'bicycle-sharing'.

4.1 Anàlisis del dataset

S'usaran les dades d'un Dataset de Kaggle que conté 58 estacions i 500 bicis de Seattle: https://www.kaggle.com/pronto/cycle-share-dataset?select=trip.csv.

Aquest Dataset està compost per tres arxius .csv, un amb les dades de les estacions, un amb dades dels trajectes dels usuaris i un altre amb la informació climàtica. Tot i que la informació del temps en un principi no s'usarà a l'algorisme genètic, s'analitzaran les dades que hi ha igualment ja que és possible que els dies de pluja s'extreguin del model degut a una esperada disminució de l'ús d'aquest mitjà de transport.

4.1.1 Estacions

El primer dataset que s'analitzarà és el referent a les estacions, on tindrem els següents camps:

Columna	Descripció	Tipus	Exemple
station_id	Id únic de l'estació	str	BT-01
name	Nom de l'estació	str	3rd Ave & Broad St
lat	Latitud on es troba	float	47.618418
long	Longitud on es troba	float	-122.350964
install_date	Dia que comença a usar-se	date	10/13/2014
install_dockcount	Número d'espais (docks) per bicis	int	18
modification_date	Modificació de localització o dock	date	10/13/2014
current_dockcount	Número de docks 31/08/2016	int	18
decommission_date	Dia de baixa de l'estació o punt	date	10/13/2014

Taula 4: Dataset de les estacions de bici

Trobem un dataset amb 58 registres on veiem que és una espècie de taula mestre de les estacions. Mirem si tots els camps estan informats en tots els registres:

```
station_id            0
name                  0
lat                   0
long                  0
install_date          0
install_dockcount     0
modification_date    41
current_dockcount     0
decommission_date    54
```

Tots els camps venen informats menys els punts que no han patit cap modificació i els 54 punts que encara no han estat donats de baixa, per tant tenim que hi ha 4 punts que no són operatius a dia 31/08/2016.

Utilitzarem en aquest anàlisi algunes de les gràfiques que ens proporciona la pròpia pàgina de Kaggle ja que hi podem obtenir bona informació de la distribució de les dades.

Pel que fa a la data d'instal·lació, quasi tots els punts de bicis van ser instal·lats el dia 13/09/2019:

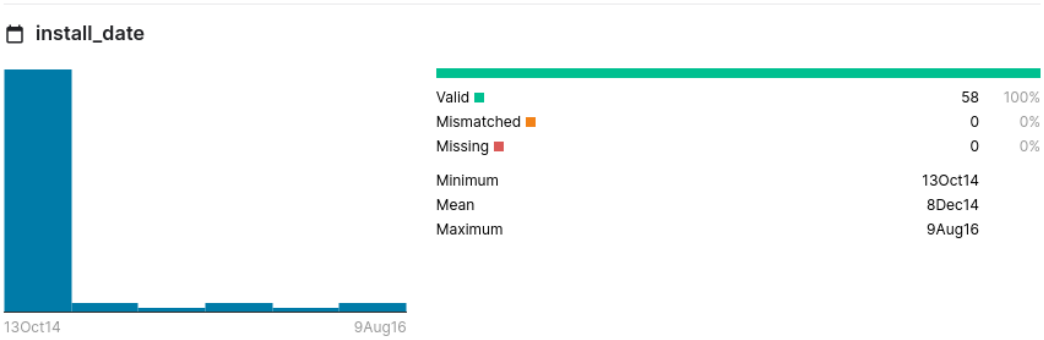

Figura 13: Resum del camp install_date, font:https://www.kaggle.com/pronto/cycle-share-dataset/version/2

Mentre que les dates de baixa dels 4 docks van ser entre finals del 2015 i 2016:

```
{'10/29/2015', '3/18/2016', '7/2/2016', '8/9/2016', nan}
```

S'ha de vigilar el format de les dates que al provenir d'EEUU tenen el format mm/dd/aaaa.

El número de docks o punts per deixar les bicis sembla que segueix una normal on la mitjana es situaria entre els 16 i els 17 llocs amb una desviació estàndard de 5:

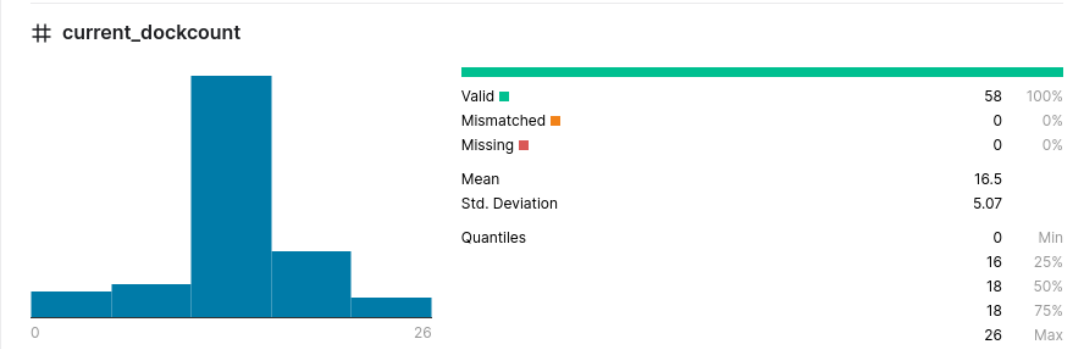

Figura 14: Resum del camp current_dockcount, font:https://www.kaggle.com/pronto/cycle-share-dataset/version/2

Finalment, podem fer una representació dels punts a l'espai aprofitant els camps de longitud i latitud. Podem afegir un camp extra que representi si el punt té pocs docks, els docks normals o molts docks. Aquest camp extra el realitzem amb una aproximació molt 'naive' fent servir com a punts de tall la mitjana +- la desviació estàndard, si està per sota del valor direm que té pocs llocs, si està per sobre que en té molts i si està entre els intervals direm que és normal:

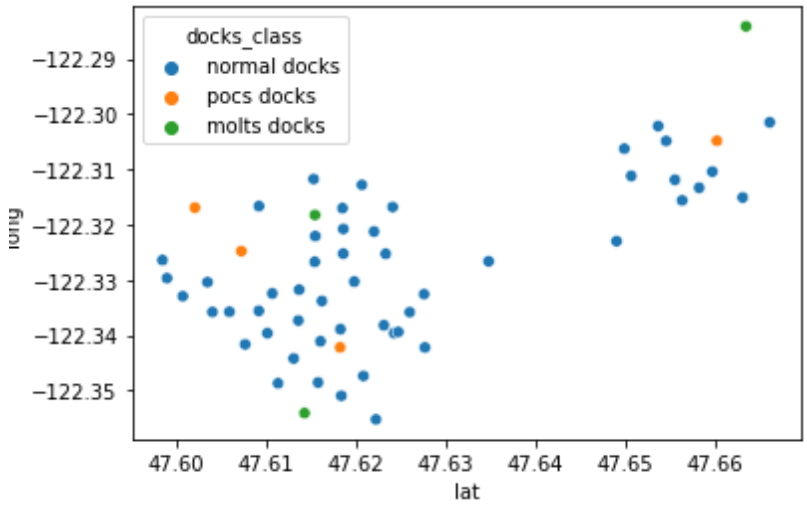

Figura 15: Estacions de bici situades per longitut i latitut

Més endavant en el procés de desenvolupament de l'algorisme s'estudiarà si és un factor important i s'ha de realitzar alguna mena de clustering que agrupi tipus de llocs tal i com s'ha fet a l'estudi de Lahoorpoor [18]. Si treiem el que s'han cancel·lat veiem que eren els que ja tenien pocs docks:

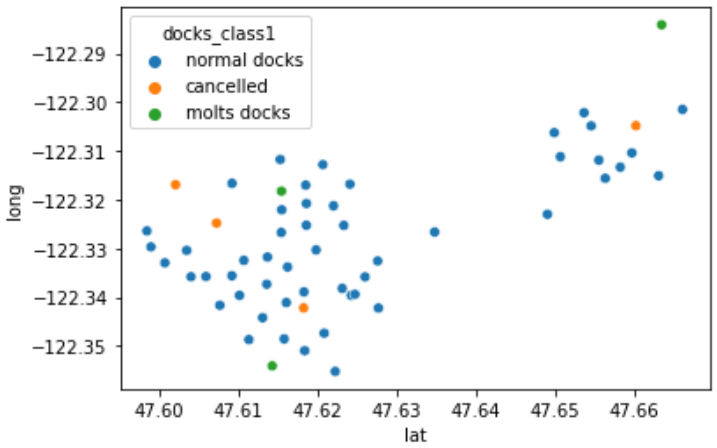

Figura 16: Estacions de bici cancel·lades

Veiem que hi ha dues zones clarament diferenciades, probablement la zona central sigui la que ocupa el Green Lake:

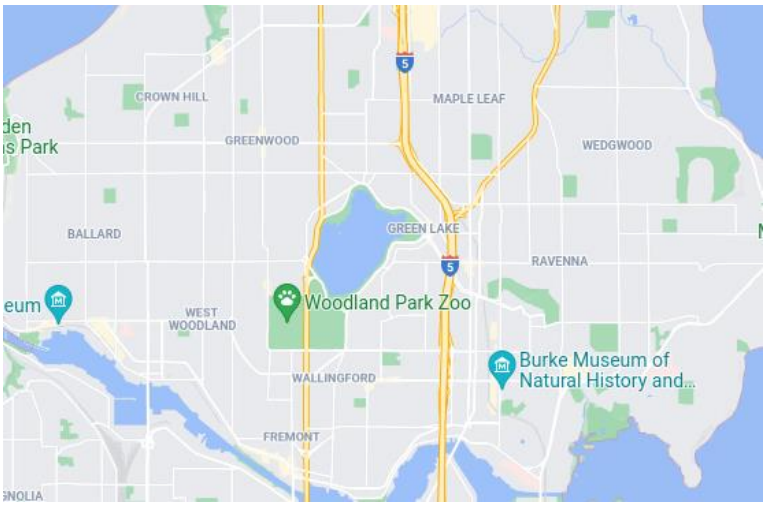

Figura 17: Mapa de la zona d'estudi

4.1.2 Viatges ‘trips’

Aquest segon dataset és el més important per poder aplicar l’algorisme genètic i són tots els viatge que han fet els usuaris entre estacions amb les bicis. Té un total de 286.859 viatges, veiem les seves columnes:

Columna	Descripció	Tipus	Exemple
trip_id	Id únic del viatge	Int	431
starttime	Dia i hora a l’iniciar el viatge	Time	10/13/2014 10:31
stoptime	Dia i hora al finalitzar el viatge	Time	10/13/2014 10:41
bikeid	Identidficador únic de la bici	Str	SEA00298

tripduration	Duració del viatge	float	985.935
fromstationname	Nom de l'estació origen	Str	2nd Ave & Spring
tostationname	Nom de l'estació destí	Str	Occidental Park
fromstationid	Id únic de l'estació origen	Str	CBD-06
tostationid	Id únic de l'estació destí	Str	PS-04
usertype	Tipus d'usuari	Str	Member
gender	Genere de l'usuari	Str	Male
birthyear	Any de naixement de l'usuari	Int	1960

Taula 5: Dataset trips

Al carregar les dades veiem que hi ha un viatge que no té ben informades les dades. El problema és que a continuació de les dades s'han tornat a introduir els noms dels camps de la taula:

```
50794 59000,"4/17/2015 14:21","4/17/2015 19:21","SEA00362",17990.668,"6th Ave S & S King St","Westlake Ave & 6th Ave","ID-04","SLU-
      15"trip_id","starttime","stoptime","bikeid","tripduration","from_station_name","to_station_name","from_station_id","to_statio
      n_id","usertype","gender","birthyear"
```

S'ha optat per descartar el registre a l'hora de carregar les dades i perdre el viatge. Veiem que totes les dades estan informades menys el gènere dels usuaris i l'any de naixement:

```
trip_id                 0
starttime               0
stoptime                0
bikeid                  0
tripduration            0
from_station_name       0
to_station_name         0
from_station_id         0
to_station_id           0
usertype                0
gender             105300
birthyear          105304
```

En aquest dataset trobem informació que podria ser important com el tipus d'usuari, ja que els usuaris que són membres és probable que tingui un patró d'us de les bicis que es repeteixi en el temps i sigui més fàcil calcular la capacitat dels punts en funció d'aquests usuaris. També es pot mirar d'analitzar si hi ha punts de bicis que tenen una proporció més gran d'usuaris no habituals, per exemple. Aquesta informació es podria mirar d'utilitzar si finalment s'opta per fer un clustering d'estacions com s'ha mencionat en el punt anterior. Veiem que un 63% dels viatges els han realitzat usuaris membres:

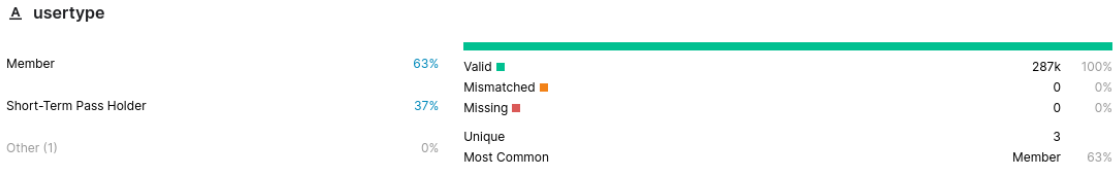

Figura 18: Resum d'usertype, font:https://www.kaggle.com/pronto/cycle-share-dataset/version/2 4

Veiem quines són les 10 estacions amb més viatges rebuts:

```
to_station_name
2nd Ave & Pine St                                  13784
Pier 69 / Alaskan Way & Clay St                    13736
Westlake Ave & 6th Ave                             10961
3rd Ave & Broad St                                 10737
PATH / 9th Ave & Westlake Ave                      10632
Occidental Park / Occidental Ave S & S Washington St    9584
Republican St & Westlake Ave N                      9305
Pine St & 9th Ave                                   9114
Seattle Aquarium / Alaskan Way S & Elliott Bay Trail    8931
1st Ave & Marion St                                 8713
```

I les 10 estacions de les que surten més usuaris:

```
Pier 69 / Alaskan Way & Clay St          13054
E Pine St & 16th Ave                     11392
3rd Ave & Broad St                       10934
2nd Ave & Pine St                        10049
Westlake Ave & 6th Ave                    9994
E Harrison St & Broadway Ave E            9639
Cal Anderson Park / 11th Ave & Pine St    9468
REI / Yale Ave N & John St                8382
2nd Ave & Vine St                         8168
15th Ave E & E Thomas St                  7680
```

Hi ha varies que es repeteixen, pel que podríem dir que són estacions molt concorregudes. Si agafem les estacions on més bicis es deixen veiem que algunes s'han ampliat el número de docks i que totes tenen entre 18 i 24 docks:

to_station_name	trip_id	install_dockcount	current_dockcount	docks_class
Pier 69 / Alaskan Way & Clay St	9385	18	24	molts docks
PATH / 9th Ave & Westlake Ave	8954	18	18	normal docks
2nd Ave & Pine St	7986	18	18	normal docks
Republican St & Westlake Ave N	7836	18	18	normal docks
Westlake Ave & 6th Ave	7804	12	20	normal docks
Pine St & 9th Ave	7541	20	20	normal docks
Seattle Aquarium / Alaskan Way S & Elliott Bay...	7331	18	18	normal docks
REI / Yale Ave N & John St	7004	18	20	normal docks
3rd Ave & Broad St	6508	18	18	normal docks
2nd Ave & Spring St	6206	20	18	normal docks

Figura 19: Estacions amb més viatges finalitzats

Podem realitzar un anàlisi similar però analitzant per les estacions d'on més bicis s'agafen quin tipus d'usuaris ho fan i veiem que hi ha algunes que són més usades per membres, mentre que altres tenen un comportament més d'usuaris ocasionals:

from_station_name	usertype	
2nd Ave & Pine St	Member	5475
	Short-Term Pass Holder	4574
3rd Ave & Broad St	Member	4274
	Short-Term Pass Holder	6660
E Pine St & 16th Ave	Member	9895
	Short-Term Pass Holder	1497
Pier 69 / Alaskan Way & Clay St	Member	3891
	Short-Term Pass Holder	9163
Westlake Ave & 6th Ave	Member	6795
	Short-Term Pass Holder	3199

Figura 20: Tipus d'usuaris de les estacions amb més viatges iniciats

Si mirem la duració dels viatges veiem que la informació està en segons i que hi ha uns pocs que han tingut una durada molt gran de més d'una hora:

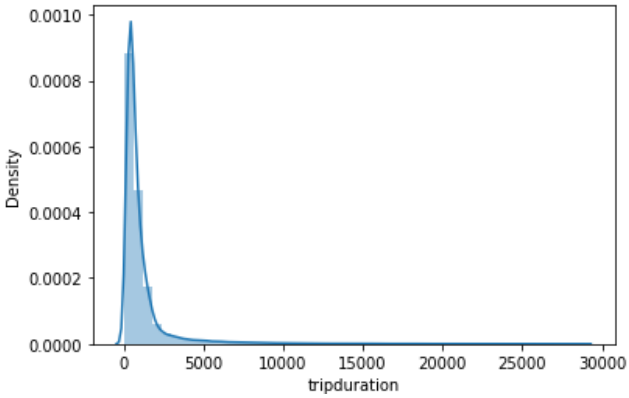

Figura 21: Duració dels viatges en segons

Si traiem aquests 'outliers' de duració i ens quedem amb trajectes de fins a una hora, veiem que la majoria de viatges han durat uns 10 minuts:

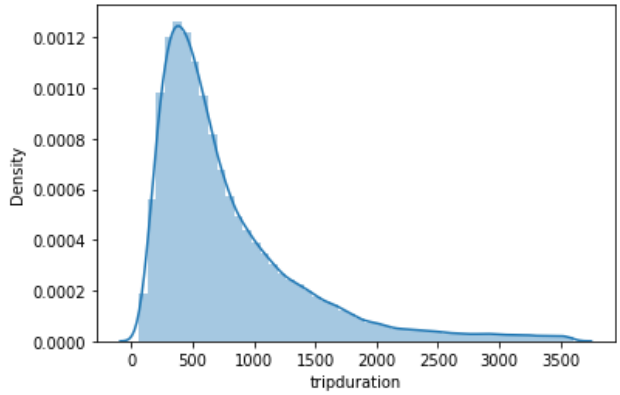

Figura 22: Duració dels viatges sense ouliers

Si extraiem les hores del dia a la que s'han realitzat els viatges veiem que pel matí entre les 8 i les 10 hi ha més demanda, s'estabilitza fins les 17 de la tarda i que després va decaient el ús fins a ser mínim de nit i matinada:

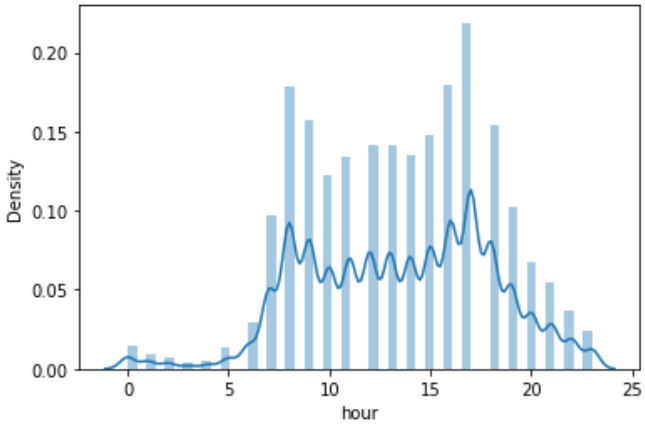

Figura 23: Viatges per hora del dia

Finalment analitzem els dies i veiem que hi ha un component estacional, on hi ha un descens de viatge s ales èpoques d'hivern, finals de 2014 i principis de 2015 i finals de 2015 i principis de 2016, segurament degut al fred o a la pluja:

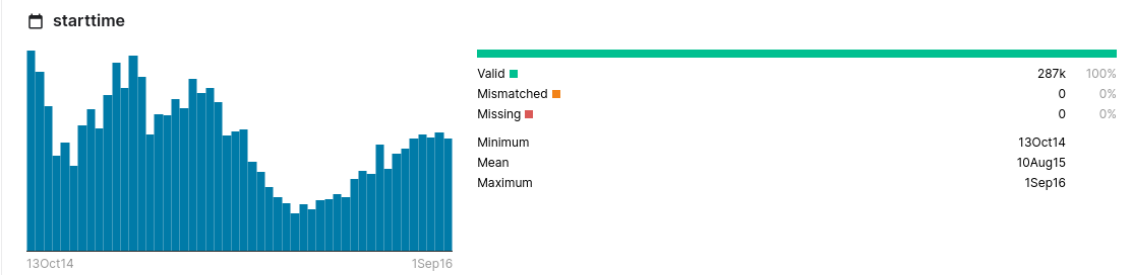

Figura 24: Resum del camp starttime, font:https://www.kaggle.com/pronto/cycle-share-dataset/version/2

4.1.2 Temps

Finalment analitzem el dataset del temps i veiem que hi ha tres factors, els valors de temperatura, humitat i dew_point (temperatura a la que l¡'aire s'ha de refredar per saturar-se amb vapor d'aigua):

Columna	Descripció	Tipus	Exemple
date	Dia de l'any	date	10/13/2014
max_temperture	Màxima temperatura	int	71
mean_temperature	Mitjana de temperatura	int	62
min_temperature	Mínima temperatura	int	54
max_dew_point_F	Màxim dew_point	int	55
mean_dew_point_F	Mitjana de dew_point	int	51
min_dew_point_F	Mínim dew_point	int	48
max_humifity	Màxima humitat	int	84
mea_humidity	Mitjana de humitat	int	80
min_humidity	Mínima humitat	int	75

Taula 6: Dataset de temps

Hagués estat útil tenir un indicador de pluja o els litres per metres quadrat, per exemple, ja que és un factor prou important a considerar per no agafar un transport com la bici. Amb els camps que hi ha disponibles en principi no es farà cap estudi per inferir si es un dia advers per agafar la bici o no.
Tot i així veiem algunes distribucions que ens proporciona Kaggle per entendre una mica per sobre el temps que ens podem trobar a Seattle.

La temperatura la trobem en mesura de Farhenheit, per passar-la a Celsius s'ha de restar 32 unitats i dividir-la entre 0,55. Veiem que poden arribar a tenir tenen temperatures altes, però en general tenen un clima bastant moderat entre 18-23 ºC. Poden arribar a estar al voltant de 0ºC a l'hivern però no és l'habitual:

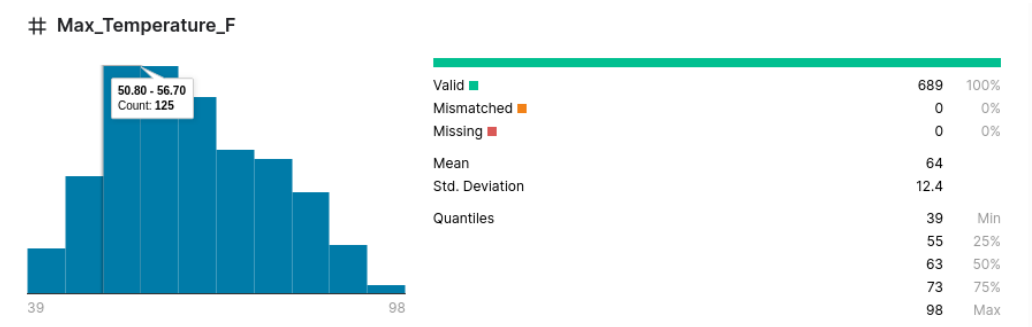

Figura 25Màxim Dades de temperatura, font: https://www.kaggle.com/pronto/cycle-share-dataset/version/2

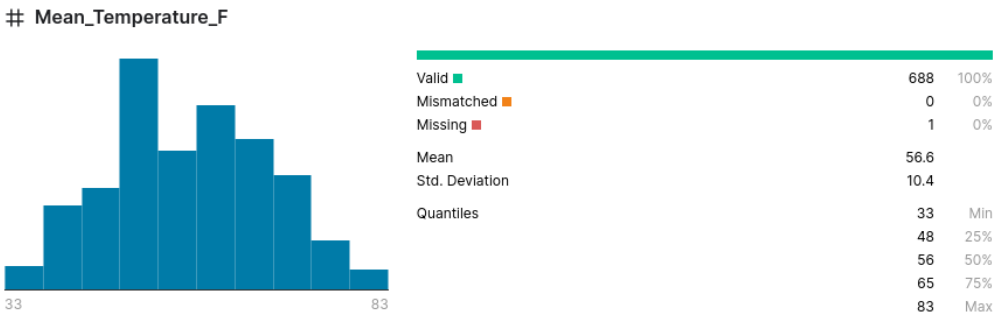

Figura 26: Mitjana de temperatures, font: https://www.kaggle.com/pronto/cycle-share-dataset/version/2

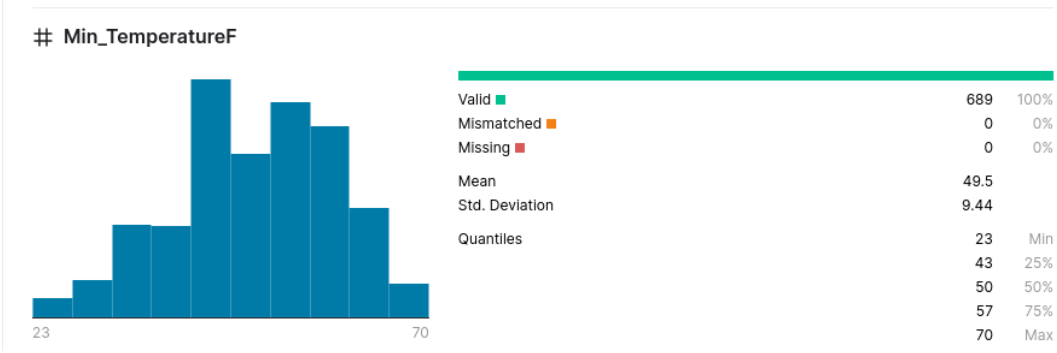

Figura 27: Mínim de temperatura, font:https://www.kaggle.com/pronto/cycle-share-dataset/version/2

La variable dew_point està molt relacionada amb la humitat, pel que analitzarem ala humitat mitja de la ciutat per fer-nos una idea de com és. Veiem que es tracta d'una ciutat humida amb una humitat relativa entre el 60 i el 80%, similar al nord d'Espanya:

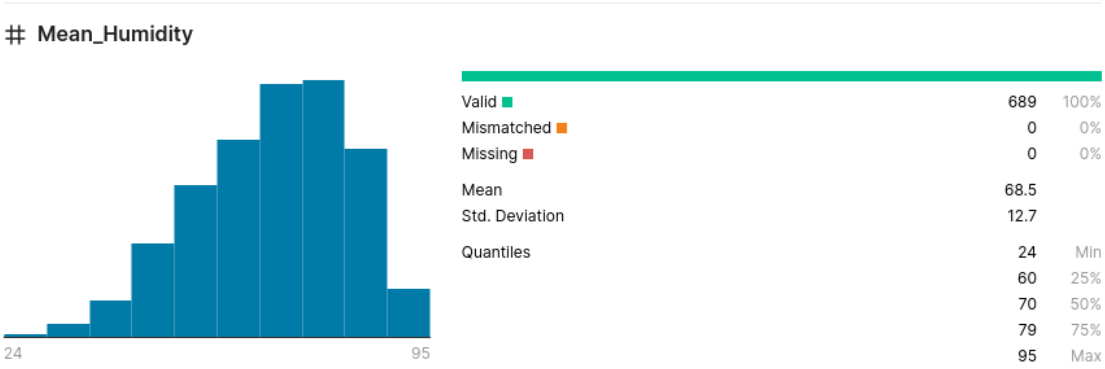

Figura 28: Mitjana d'humitat, font:https://www.kaggle.com/pronto/cycle-share-dataset/version/2

4.2 Definició de l'algorisme genètic aplicat

Tal i com s'ha vist a l'apartat 2.1, els problemes d'optimització es basen en obtenir la millor solució possible per a minimitzar o maximitzar una funció. En aquest cas el problema que es vol minimitzar és, donat uns punts de servei de bike-sharing i un nombre de bicis i espais disponibles, obtenir el mínim nombre d'usuaris que no han pogut fer ús del servei, ja sigui perquè no hi havia bicis en el punt o perquè no l'han pogut retornar. La funció a minimitzar doncs seria gairebé idèntica a la de l'article treballat a l'estat de l'art [5]:

min f (x) = E x [#failed-starts] + E x [#failed-ends]

on, els failed-starts són els viatges que no s'han pogut realitzar perquè no hi havia bicis disponibles en aquell punt i els failed-ends els que l'usuari no ha pogut deixar la bici perquè estaven tots els punts ocupats.

Pel que fa a les restriccions del model són les següents:

- **número de bicis total màxim <= 500**
- **12 >= número de docks <= 24**
- **6 >= número de bicis al punt <= 12**
- **1>= Dia de la setmana <= 5**

S'ha limitat el nombre d'espais (docks) i de bicis perquè no té sentit que hi hagin punts enormes amb moltes bicis i espais lliures dintre d'una ciutat. Aquests punts donarien una funció objectiu molt baixa o 0 però no són viables, a més de fer el problema gairebé intractable. Els valors que s'han fet servir han estat basats en el nombre mínim i màxim que hi ha a les estacions del dataset inicial. Finalment s'ha decidit optar per treballar només amb els viatges de dilluns a divendres ja que són típicament els dies laborables i, per tant, els dies que els viatges són més 'rutinaris' i per tant en els que el model es pot ajustar millor amb una demanda similar al llarg dels dies, tal i com veiem en el nombre de viatges per dia a l'apartat 4.3.2.

Tal i com veurem més endavant amb la definició dels cromosomes i de les funcions de selecció i mutació, s'ha desenvolupat un algorisme heurístic que tractarà de resoldre el problema donant una solució aproximada.

4.2.1 Cromosomes

Els cromosomes tindran 54 gens, un per cada estació, i aquests tindran dos al·lels, tal i com ja es feia a l'article [11] vist a l'estat de l'art, un amb la informació dels espais del punt i l'altre les bicis que hi ha. Per tant cada individu podrà emmagatzemar la informació necessària de cada estació o punt de servei.

Les posicions dels gens doncs seran fixes, fent referència cada posició a una estació concreta, com a l'article [16], on el primer valor del gen (primer al·lel) emmagatzemarà el número total de 'docks' del punt de servei i el segon emmagatzemarà el nombre de bicis que hi haurà a l'inici del dia.

Pel que fa al nombre total de cromosomes de la població s'executarà diferents proves per obtenir un òptim (N) tenint en compte els resultats obtinguts i els temps d'execucions.

Per tant tindrem:

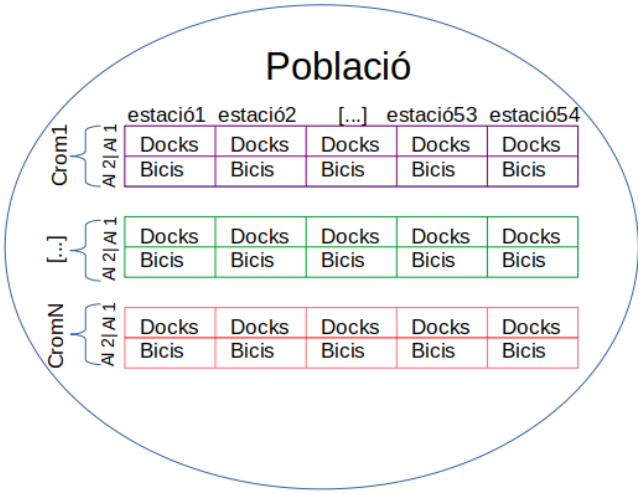

Figura 29: Població de cromosomes per l'AG

Es crea una població inicial de N cromosomes amb valors aleatoris pels seus 54 gens. Tal i com s'ha vist a les restriccions del model, els valors de cada gen a a l'al·lel 1, nombre d'espais del punt de bici, serà un número entre 12 i 24, mentre que a l'al·lel 2, nombre de bicis al punt, els valors seran entre 6 i 12. Addicionalment, la suma de l'al·lel 2 de cada cromosoma no podrà ser superior a 500, és a dir no podrà haver cap solució que tingui més de 500 bicis a totes les estacions.

L'algorisme genètic avalua per generació els cromosomes en un període de temps que es delimiti. Durant aquest període, a l'inici de cada dia del període cada estació tindrà la configuració que s'ha establert i s'anirà actualitzant el valor de l'al·lel 2 conforme vagin passant les hores en funció del nombre d'usuaris que agafin o deixin bicis. El valor de l'al·lel 2 de cada gen mai podrà ser inferior a 0 ni superior al valor de l'al·lel 1, ja que no poden haver bicis en valor negatiu ni més bicis que espais disponibles. En el moment que un usuari no pugui agafar una bici perquè el valor sigui 0 o no pugui deixar la bici perquè el valor sigui igual al de l'al·lel 1 la funció objectiu augmentarà i el valor de l'al·lel 2 es mantindrà igual perquè aquell trajecte no s'ha pogut realitzar. Al final del dia, els valors de l'al·lel 2 tornaran al valor inicial de la configuració (s'entén que hi ha un procés

nocturn de càrrega i descàrrega de punts de bici per deixar-los en l'estat que es vol avaluar).

Un cop finalitzi la generació i es seleccionin els millors individus, veure apartat 4.4, aquests es creuaran. Els dos al·lels d'un mateix gen es consideren indivisibles, és a dir que en el procés d'encreuament s'intercanviaran els dos al·lels entre cromosomes. S'ha establert un procés d'encreuament de 4 punts aleatoris, per tant, en cada parella de cromosomes podran ser diferents els gens (estacions) que s'intercanviïn. Quan es produeix l'encreuament per a cada parella de cromosomes creuats es generen 2 nous cromosomes. Veiem un exemple d'encreuament en 4 punts en que s'han seleccionat els gens 6, 8, 23, 42 aleatòriament:

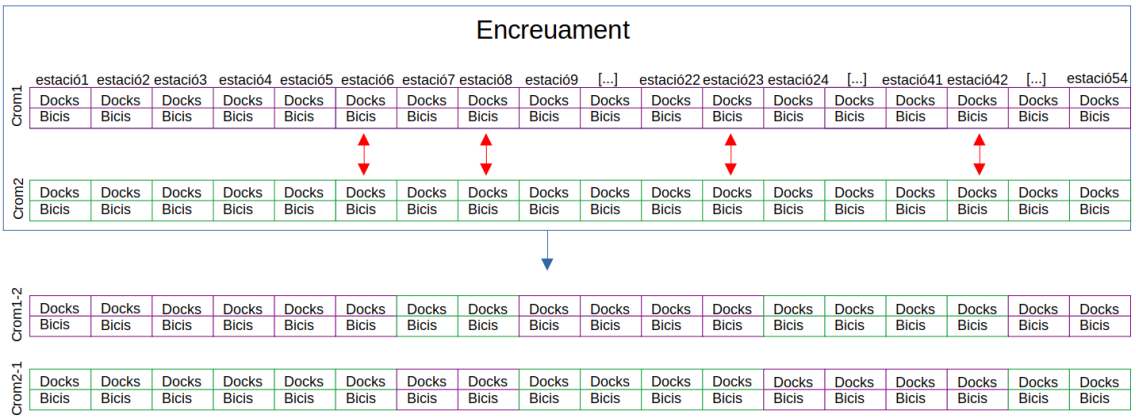

Figura 30: Exemple d'encreuament entre dos cromosomes

Finalment, també de forma aleatòria, es produeix el fenomen de mutació amb una probabilitat del 10%, per tant, cada cromosoma nou que es genera té un 10% de probabilitats de mutar en aquella generació. S'ha establert que si el cromosoma muta ho faci en 4 gens aleatoris per a que hi hagi més variabilitat. El gen muta en els dos al·lels, i rep un valor nou en cada un d'ells dintre del rang establert en les restriccions de forma aleatòria. Veiem els cromosomes nous creats a la imatge anterior com quedarien en el cas que el segon cromosoma mutés i ho fes en els gens 2, 22, 41, 54:

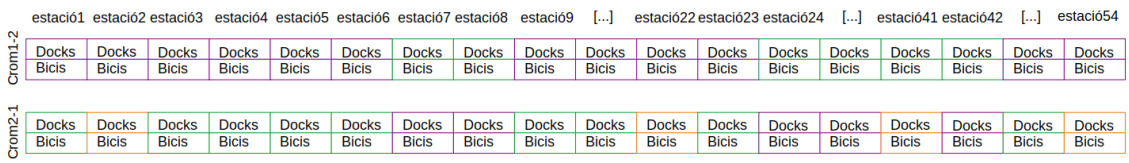

Figura 31: Exemple de mutació de dos cromosomes

Finalment l'algoritme valida que tots els cromosomes siguin vàlids, en aquest cas com la restricció del nombre de espais al dock i el nombre de bicis per estació ja

està en el propi procès, el que es valida és el nombre de bicis totals per cromosoma. Si hi ha un cromosoma que no compleix la restricció es considera com no-viable i no continua a la següent generació. Per mantenir la mida de la població es torna a encreuar els millor cromosomes fins a obtenir el mateix nombre de població.

4.3 Tractament de les dades

Després de veure la complexitat que pot tenir aquest model en que s'intentarà trobar l'òptim per les 54 estacions, s'ha decidit no incloure el dataset del Temps, apartat 4.1.2, en la creació del model. El factor que suposadament més podria afectar es la pluja, ja que faria que menys usuaris agafessin la bicicleta, però aquest factor no queda clar amb les dades que es tenen.

Per tant s'utilitzaran les dades dels datasets:

- Estacions
- Viatges

S'han carregat les dades dels dos datasets a partir dels csv descarregats de la pàgina de Kaggle: https://www.kaggle.com/pronto/cycle-share-dataset/version/2?select=station.csv

El dataset d'estacions s'ha pogut carregar sense problemes mentre que en el dataset de viatges, tal i com es va veure a l'apartat 4.1.2, s'ha hagut d'eliminar una línia errònia.

4.3.1 Estacions

En aquest dataset l'única transformació que s'hi ha realitzat és convertir la data d'instal·lació en format Date ja que al carregar les dades es trobava en format 'string':

```
df_stations['datetime_install'] = pd.to_datetime(df_stations['install_date'])
```

4.3.2 Viatges

Dataset principal en el que hi ha tots els viatges. En aquest cas s'han hagut de transformar les variables d'inici i finalització de viatge a tipus Date, i s'ha extret com a columnes extres la informació de l'any, mes, dia, i dia de la setmana de forma separada per la posterior implementació del model:

```
df_trips['datetime_start'] = pd.to_datetime(df_trips['starttime'])
df_trips['datetime_stop'] = pd.to_datetime(df_trips['stoptime'])
df_trips['year'] = df_trips.apply(lambda row : row['datetime_start'].year, axis = 1)
df_trips['month'] = df_trips.apply(lambda row : row['datetime_start'].month, axis = 1)
df_trips['day'] = df_trips.apply(lambda row : row['datetime_start'].day, axis = 1)
df_trips['weekday'] = df_trips.apply(lambda row : row['datetime_start'].weekday(), axis = 1)
```

Analitzem de forma breu com es distribueixen els viatges en funció de l'any i del mes:

year	month	
2014	10	13170
	11	15646
	12	11662
2015	1	14736
	2	14660
	3	19960
	4	18773
	5	15548
	6	15999
	7	18808
	8	17046
	9	13134
	10	10605
	11	6541
	12	5049
2016	1	5162
	2	5786
	3	6973
	4	9029
	5	10487
	6	11548
	7	13342
	8	13193

I en funció del dia de la setmana:

weekday	
0	42204
1	42886
2	42941
3	43946
4	43300
5	38472
6	33108

Veiem que efectivament el nombre de viatges de dilluns a divendres es similar, mentre que el cap de setmana es comporta diferent. Per tant, ens quedarem tant sols amb els viatges de dilluns a divendres:

```
df_trips_workdays = df_trips[df_trips['weekday'] < 5]
```

Per tal de simplificar l'algorisme i degut a que es busca optimitzar sobretot els viatges regulars, no es tindran en compte el minut exacte en el que l'usuari ha agafat la bici:

```
df_trips_workdays['hour_start'] = df_trips_workdays.apply(lambda row : row['datetime_start'].hour, axis = 1)
df_trips_workdays['hour_stop'] = df_trips_workdays.apply(lambda row : row['datetime_stop'].hour, axis = 1)
```

Finalment, seguint la lògica d'un dia laborable la primera hora, moment en el que cada punt de servei ha de tenir les bicis necessàries per la resta del dia, la establirem a les 7 del matí. Es necessari doncs realitzar un mapeig per tal de poder ordenar les hores dels viatges en base a això ja que sino es començaria a contar a partir de les 00.00 de la nit:

```
df_mapping = pd.DataFrame({
    'hour_order': [7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,0,1,2,3,4,5,6],
})

sort_mapping = df_mapping.reset_index().set_index('hour_order')

df_trips_workdays['hour_order'] = df_trips_workdays['hour_start'].map(sort_mapping['index'])
```

4.4 Creació del model d'optimització

El model s'ha implementat, tal i com s'ha vist a l'apartat 2.2.2, amb els 4 passos de tot algorisme genètic ha de tenir, iniciació, selecció, operadors genètics i finalització.

- Iniciació

El model genera una població inicial de forma aleatòria composta per un Nombre N de cromosomes, on cada cromosoma estarà compost per 54 gens (nombre total d'estacions) amb dos al·lels, el nombre de docks i de bicis.

S'han definit 3 funcions per executar la part d'iniciació. La primera per obtenir el nombre de bicis de cada gen, on se li aplica la restricció del nombre mínim i màxim de bicis per estació i el nombre total de bicis:

```
def bikes_dock(min_bikes, max_bikes, sum_bikes):
    bikes = np.random.randint(min_bikes, max_bikes, df_stations.shape[0])
    while (sum(bikes) > sum_bikes):
        bikes = np.random.randint(min_bikes, max_bikes, df_stations.shape[0])
    return bikes
```

Una segona funció que genera un cromosoma amb els dos al·lels, per un costat el nombre de docks seguint les restriccions de mínim i màxim, i per un altre costat el nombre de bicis fent ús de la funció anterior:

```
def cromosome(stations, min_docks, max_docks, min_bikes, max_bikes, sum_bikes):
    stations['current_dockcount'] = np.random.randint(min_docks, max_docks, stations.shape[0])
    stations['bikes'] = bikes_dock(min_bikes, max_bikes, sum_bikes)
    stations['bikes'] = np.where(stations['bikes'] > stations['current_dockcount'], stations['current_dockcount'],sta
    return df_stations
```

Finalment, una tercera funció que executa les dues anteriors per obtenir una població de N cromosomes (num_population):

```python
def initiation(stations, num_population,  min_docks=12, max_docks=24, min_bikes=6, max_bikes=12, sum_bikes=500):
    population = []
    for i in range(num_population):
        cromosoma = cromosome(df_stations,  min_docks, max_docks, min_bikes, max_bikes, sum_bikes)
        cromosoma = cromosoma[['station_id', 'current_dockcount', 'bikes']]
        population.append(cromosoma)

    return population
```

- Selecció

Per poder fer la selecció dels millors cromosomes s'ha de calcular per a cada cromosoma la funció objectiu a totes les estacions. La forma de calcular la funció objectiu serà sumant el nombre de bicis que no es poden agafar en a l'estació en aquell moment, ja que hi ha més usuaris que inicien viatges (datetime_start) que bicis en el dock, amb el nombre d'usuaris que no poden deixar la bici ja que hi ha més usuaris que finalitzen en aquella estació (datetime_stop) que espais lliures. No es té en compte el nombre de bicis que es deixen o s'agafen en aquella hora en el càlcul ja que tal i com s'ha plantejat l'algorisme no se sap quins viatges han estat abans o després.

S'ha creat una funció auxiliar que donat un valor de funció objectiu i una estació, calcula el resultat de la funció objectiu i el suma per obtenir el valor total.

```python
def station_day(id_st, objective, take_bikes, left_bikes, df_stations):
    dockcount = df_stations[df_stations['station_id'] == id_st]['current_dockcount'].values
    bikes_dock = df_stations[df_stations['station_id'] == id_st]['bikes'].values
    free_docks = dockcount - bikes_dock

    for hour in range(24):
        take = take_bikes[(take_bikes['from_station_id'] == id_st) & (take_bikes['hour_order'] == hour)]['trip_id'].v
        left = left_bikes[(left_bikes['to_station_id'] == id_st) & (left_bikes['hour_order'] == hour)]['trip_id'].val

        if take.size == 0:
            take = 0
        if left.size == 0:
            left = 0

        objective = objective + (0 if bikes_dock >= take else take - bikes_dock) + (0 if free_docks >= left else left

        bikes_dock = bikes_dock - take + left
        if (bikes_dock < 0):
            bikes_dock = 0
        elif(bikes_dock > dockcount):
            bikes_dock = dockcount
        free_docks = dockcount - bikes_dock

    return objective
```

Amb la funció anterior i donades unes dates, ja es pot calcular el resultat de la variable objectiu per a tots els cromosomes de la població per a tots els dies entre les dates. Tal i com s'ha vist a l'apartat 4.3.2 s'ha definit les 7 del matí com primera hora, en aquell moment els docks i les bicis que hi hauran a cada estació són els que estan definits per l'algorisme. Cada hora s'actualitzen el nombre de bicis de l'estació en funció del nombre de usuaris que agafen o deixen bicis en els docks i es calcula la funció objectiu al llarg del dia i es va acumulant. Cada dia del bucle a les 7 del matí es torna a tenir els valors inicials per a cada estació:

```
def new_generation(num_population, population, initial_date, days_month):
    population_obj = []

    for cromosoma in range(num_population):
        df_stations = population[cromosoma]
        objective_function = 0

        for single_date in (pd.to_datetime(initial_date) + timedelta(n) for n in range(days_month)):
            year = single_date.year
            month = single_date.month
            day = single_date.day

            trip_day = df_trips_workdays[(df_trips_workdays['year'] == year) & (df_trips_workdays['month'] == month) &
            trip_day = trip_day[['trip_id', 'datetime_start', 'datetime_stop', 'from_station_id', 'to_station_id', 'h

            one_day_trip_take = trip_day[['trip_id', 'from_station_id', 'datetime_start', 'hour_start', 'hour_order']
                        .merge(df_stations[['station_id', 'current_dockcount', 'bikes']], \
                               how='left', left_on='from_station_id', right_on='station_id')\
                        .sort_values(by = ['station_id', 'hour_order'])

            one_day_trip_left = trip_day[['trip_id', 'to_station_id', 'datetime_start', 'hour_start', 'hour_order']]\
                        .merge(df_stations[['station_id', 'current_dockcount', 'bikes']], \
                               how='left', left_on='to_station_id', right_on='station_id')\
                        .sort_values(by = ['station_id', 'hour_order'])

            take_bikes = one_day_trip_take.groupby(['from_station_id', 'hour_start', 'hour_order', 'current_dockcount

            left_bikes = one_day_trip_left.groupby(['to_station_id', 'hour_start', 'hour_order', 'current_dockcount']

            for station in set(df_stations['station_id']):
                objective_function = station_day(station, objective_function, take_bikes, left_bikes, df_stations)
        population_obj.append(objective_function)

    return population_obj
```

S'han implementat dues funcions diferents per fer la selecció dels millors individus un cop tenim el resultat de la funció objectiu, una de selecció tipus rank i l'altre per truncat, veure apartat 2.2.2. S'ha decidit utilitzar la selecció de tipus rank a l'algorisme ja que no hi ha molta diferència entre els valors de fitness i volem que hi hagi més variabilitat en la selecció, tal i com s'ha vist a l'apartat 2.2.2. La selecció de tipus truncat s'usarà per comparar resultats entre els dos mètodes:

```
def selection_trunc(population, population_obj_fun, num_crom_best):
    population_to_order = population_obj_fun
    population_to_order = sorted(population_to_order)

    best_cromosomes = []
    best_obj_fun = []
    for i in range(num_crom_best):
        value = population_to_order[i]
        best_obj_fun.append(value)
        index = population_obj_fun.index(value)
        best_cromosomes.append(population[index])

    return best_cromosomes, best_obj_fun

def selection_rank(population, population_obj_fun, num_crom_best):
    population_index = np.arange(num_population)
    probs = (1/ pd.Series(np.concatenate(population_obj_fun, axis=0 )).rank()) / sum(1/ pd.Series(np.concatenate(popu
    selected = np.random.choice(population_index, size = num_crom_best, replace = False, p=probs)
    best_cromosomes = []
    best_obj_fun = []
    for position in selected:
        best_cromosomes.append(population[position])
        best_obj_fun.append(population_obj_fun[position])

    return best_cromosomes, best_obj_fun
```

Veiem un gràfic comparatiu dels dos mètodes de selecció amb els resultats de l'evolució de la funció objectiu mitjana fent servir el mètode rank i una altre fent servir el mètode de truncat:

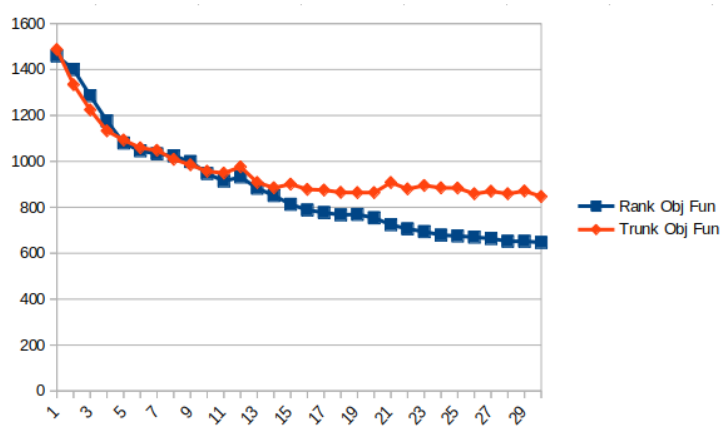

Figura 32: Comparativa de les funcions de selecció rank i trunk

Veiem que la funció de selecció truncant arriba un moment en el que ja no millora, mentre que la funció rank segueix millorant a cada generació. Això, tal i com es menciona anteriorment pot ser degut a la poca diferencia que hi ha entre funcions objectius, apartat 2.2.2.

- Operadors genètics

S'ha optat per un crossover entre parelles de cromosomes seleccionats de tipus múltiple, veure apartat 2.2.2, en el que cada parella intercanviarà cromosomes en varis punts, i aquests seran aleatoris en cada emparellament (veure apartat 4.2.1 Cromosomes):

```
def crossover_multi(best_population, num_points_cross):
    new_population = []
    for position in range(len(best_population)):

        if(position % 2 == 0):
            points = sample(range(54), num_points_cross)
            points.sort()

            new_crom = best_population[position][['station_id', 'current_dockcount']]
            new_crom['index'] = np.arange(54)
            new_crom['current_dockcount'] = np.where(new_crom['index'].between(0, points[0]), best_population[positio
                                                     (np.where(new_crom['index'].between(points[0], points[1]), best_p
                                                               np.where(new_crom['index'].between(points[1], points[2])
                                                                        np.where(new_crom['index'].between(points[2], po
                                                                                 best_population[position]['current_dockc
            new_crom['bikes'] = np.where(new_crom['index'].between(0, points[0]), best_population[position]['bikes'],
                                                     (np.where(new_crom['index'].between(points[0], points[1]), best_p
                                                               np.where(new_crom['index'].between(points[1], points[2])
                                                                        np.where(new_crom['index'].between(points[2], po
                                                                                 best_population[position]['bikes'])))))

            if (np.random.choice([0,1], size = 1, replace = True, p=[0.9, 0.1]) == 1):
                new_crom = mutation(num_mutations, min_docks, max_docks, min_bikes,max_bikes, new_crom=best_populatio

            new_population.append(new_crom[['station_id', 'current_dockcount', 'bikes']])

            new_crom = best_population[position+1][['station_id', 'current_dockcount']]
            new_crom['index'] = np.arange(54)
            new_crom['current_dockcount'] = np.where(new_crom['index'].between(0, points[0]), best_population[positio
                                                     (np.where(new_crom['index'].between(points[0], points[1]), best_p
                                                               np.where(new_crom['index'].between(points[1], points[2])
                                                                        np.where(new_crom['index'].between(points[2], po
                                                                                 best_population[position+1]['current_doc
            new_crom['bikes'] = np.where(new_crom['index'].between(0, points[0]), best_population[position+1]['bikes'
                                                     (np.where(new_crom['index'].between(points[0], points[1]), best_p
                                                               np.where(new_crom['index'].between(points[1], points[2])
                                                                        np.where(new_crom['index'].between(points[2], po
                                                                                 best_population[position+1]['bikes'])))))

            if (np.random.choice([0,1], size = 1, replace = True, p=[0.9, 0.1]) == 1):
                new_crom = mutation(num_mutations, min_docks, max_docks, min_bikes,max_bikes, new_crom=best_populatio

            new_population.append(new_crom[['station_id', 'current_dockcount', 'bikes']])

    return new_population
```

Un cop es realitza cada intercanvi de gens per crear el nou cromosoma, se li aplica una mutació amb una probabilitat del 10%. En cas de produir-se la mutació es variaran un nombre N de gens que són seleccionats de forma aleatòria (veure apartat 4.2.1 Cromosomes). Els valors que se li assignen son aleatoris dintre de les restriccions establertes per el model:

```
def mutation(num_mutations, min_docks,max_docks, min_bikes,max_bikes, new_crom):
    mutations = np.random.choice(range(54), size = num_mutations, replace = False, p=np.repeat((1/54),54))

    mutations.sort()
    for mutation in mutations:
        new_cromosoma_mut = new_crom
        new_cromosoma_mut['current_dockcount'][mutation] = sample(range(min_docks,max_docks), 1)[0]
        new_cromosoma_mut['bikes'][mutation] = sample(range(min_bikes,max_bikes), 1)[0]
    return new_cromosoma_mut
```

- Finalització

No s'ha definit un 'early stop' en funció del resultat de la funció objectiu ja que es vol veure l'evolució al llarg de N generacions.

4.4.1 Execució completa del model

El temps d'execució per cada generació són uns 1100 segons per una població de 24 cromosomes i per un mes de dades, pel que s'ha optat per fer inicialment execucions de 30 i 50 generacions en un únic mes per generació:

```
start = time.time()
population = initiation(df_stations, num_population)

population_obj_fun = new_generation(num_population, population, initial_date, days_month)

# best_population, best_objective_fun = selection_trunc(population, population_obj_fun, 10)
best_population, best_objective_fun = selection_rank(population, population_obj_fun, num_crom_best)

avg_generation.append(sum(best_objective_fun) / len(best_objective_fun))
end = time.time()

print('Generation ', 0,  ': average objective function = ', sum(best_objective_fun) / len(best_objective_fun))
print('Best objective function: ', min(best_objective_fun))
print('Number of cromosomes population: ', len(population))
print('Number of best population: ', len(best_population))
print('Time elapsed: ', end - start)

for generation in range(1, num_generations):
    start = time.time()
    new_cromosomes = crossover_multi(best_population, num_points_cross)

    population = best_population + new_cromosomes

    population_result = new_generation(num_population, population, initial_date, days_month)

    population_obj_fun = population_result

    # best_population, best_objective_fun = selection_trunc(population, population_obj_fun, 10)
    best_population, best_objective_fun = selection_rank(population, population_obj_fun, num_crom_best)

    avg_generation.append(sum(best_objective_fun) / len(best_objective_fun))

    end = time.time()
    print('Generation ', generation,  ': average objective function = ', sum(best_objective_fun) / len(best_objective
    print('Best objective function: ', min(best_objective_fun))
    print('Number of cromosomes population: ', len(population))
    print('Number of best population: ', len(best_population))
    print('Time elapsed: ', end - start)
```

Posteriorment i per tal que l'algorisme pugui generalitzar millor, s'ha realitzat una execució de 26 generacions en que el que inicialment s'executa el mes de maig

durant 6 generacions i, després, s'executen els mesos de gener a maig en rondes de 4 generacions. La funció objectiu s'ha hagut de dividir entre el nombre mensual de desplaçaments ja que la funció objectiu calcula valors totals per poder normalitzar el valor:

```
for generation in range(6):
    start = time.time()
    new_cromosomes = crossover_multi(best_population, num_points_cross)

    population = best_population + new_cromosomes

    month = 5
    initial_date = str(year)+'-'+ str(month).zfill(2)+'-01'
    days_month = monthrange(int(initial_date[0:4]), int(initial_date[5:7]))[1]

    population_obj_fun, stations_results = new_generation(num_population, population, initial_date, days_month)

    # population_obj_fun = [obj_fun / trips_month[month] for obj_fun in population_obj_fun]

    # best_population, best_objective_fun = selection_trunc(population, population_obj_fun, 10)
    best_population, best_ofun = selection_rank(population, population_obj_fun, num_crom_best)

    best_objective_fun = [obj_fun / trips_month[month] for obj_fun in best_ofun]

    avg_generation.append(sum(best_objective_fun) / len(best_objective_fun))

    end = time.time()
    print('Generation ', generation,  ': Month = ', month)
    print('Average objective function = ', sum(best_objective_fun) / len(best_objective_fun))
    print('Best objective function: ', min(best_objective_fun))
    print('Time elapsed: ', end - start)

for generation in range(4, num_generations):
    start = time.time()
    new_cromosomes = crossover_multi(best_population, num_points_cross)

    population = best_population + new_cromosomes

    month = min(math.trunc(generation/4), 12)
    initial_date = str(year)+'-'+ str(month).zfill(2)+'-01'
    days_month = monthrange(int(initial_date[0:4]), int(initial_date[5:7]))[1]

    population_obj_fun, stations_results = new_generation(num_population, population, initial_date, days_month)

    #population_obj_fun = [obj_fun / trips_month[month] for obj_fun in population_obj_fun]

    # best_population, best_objective_fun = selection_trunc(population, population_obj_fun, 10)
    best_population, best_ofun = selection_rank(population, population_obj_fun, num_crom_best)

    best_objective_fun = [obj_fun / trips_month[month] for obj_fun in best_ofun]

    avg_generation.append(sum(best_objective_fun) / len(best_objective_fun))

    end = time.time()
    print('Generation ', generation,  ': Month = ', month)
    print('Average objective function = ', sum(best_objective_fun) / len(best_objective_fun))
    print('Best objective function: ', min(best_objective_fun))
```

El que s'espera amb aquesta implementació és obtenir una població que sigui una possible bona solució pel mes estudiat (maig) amb les 6 primeres generacions, per després generalitzar-ho per la resta de mesos on, a més, es tornarà a optimitzar pel mes de maig durant 4 generacions més al final del procés. Degut a una programació no gaire òptima del codi ens trobem amb algunes deficiències ja que, per un costat no s'han pogut executar tantes generacions com es voldria, i per una altre banda, la població de cromosomes no té una mida molt extens. Si es pogués executar amb una població més amplia, hi hauria una variabilitat de cromosomes més gran fent que fos més fàcil trobar el que generalitzes per a més mesos.

4.5 Avaluació del model

Per avaluar l'algorisme per un costat s'analitzarà els resultats obtinguts a l'execució i per un altre s'utilitzarà el millor resultat en unes dates diferents a mode de test.

4.5.1 Resultats obtinguts

Analitzem els resultats obtinguts en l'execució de l'algorisme pel mes de Maig en 30 i en 50 generacions veient en un gràfic l'evolució de la funció objectiu mitjana dels cromosomes a cada generació:

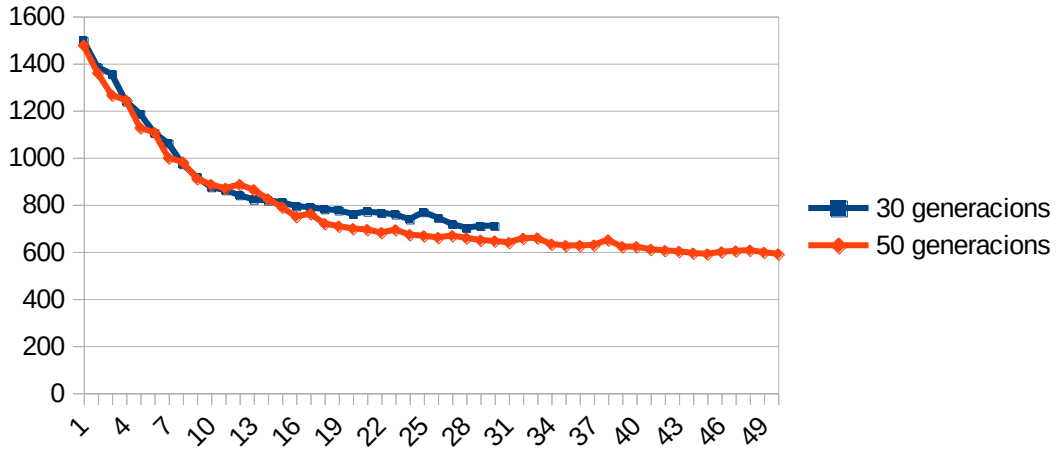

Figura 33: Evolució de la funció objectiu de l'AG en dos execucions

Veiem com al tractar-se d'un procés estocàstic en les dues execucions fins a la generació 30 tenen diferents resultats. Durant el transcurs de l'algorisme veiem que hi ha etapes en que les generacions següents no són capaces de millorar la funció objectiu, inclús l'empitjoren, però que amb el pas del temps sí que aconsegueix millorar una mica. Des de la generació 42 fins la 50 canvia poc en el temps obtenint com a millor resultat un valor de 590 a la generació 44 i a la 50. Veiem que succeeix en aquestes mateixes execucions si dibuixem el gràfic tant sols del millor cromosoma:

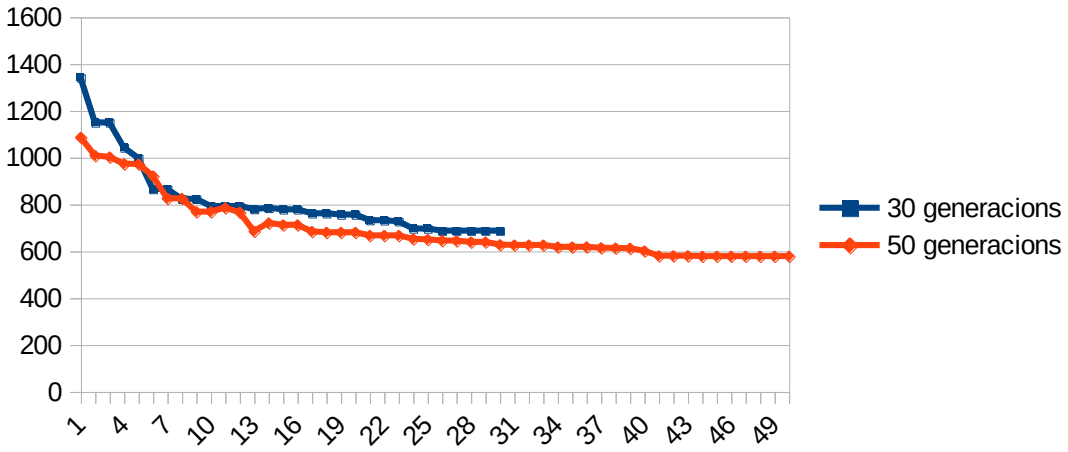

Figura 34: Evolució de la funció objectiu del millor cromosoma de cada generació

En aquest cas veiem que a la població inicial ja s'havia trobat a l'atzar un cromosoma molt millor a l'execució de 50 generacions. A partir de la generació 15 en els dos casos el pendent de la corba s'aplana molt havent-hi poques millores; si ho comparem amb el gràfic anterior veiem que la mitjana de la funció objectiu és molt pròxima al millor cromosoma, pel que s'ha trobat una població prou òptima i amb el procés d'encreuament els individus gairebé no canvien gaire i tenen resultats similars. Arribat aquest punt, és molt important el fet que hi hagi mutacions que fan que pugui aparèixer un cromosoma molt millor que els anteriors, com és el cas de la generació 12 de l'execució de 50 generacions. Finalment, podem observar com hi ha períodes en que no s'aconsegueix trobar un millor cromosoma com per exemple de la generació 44 a la 50.

Durant l'execució de l'algorisme s'emmagatzemen els resultats de la funció objectiu per a cada estació per veure si es tracta d'un resultat generalitzat i totes les estacions tenen un valor similar de funció objecti, o si pel contrari hi ha algunes estacions que tenen un comportament diferent a la resta. Primer obtenim les dades del millor cromosoma a l'execució de 50 generacions i veiem que hi ha 5 cromosomes idèntics amb el valor més baix (578). Creuem els gens del millor cromosoma amb els resultats de la funció objectiu per estació amb les de la taula mestre inicial de les estacions i amb el nombre total de viatges per estació i obtenim el següent resultat:

station_id	name	lat	long	dockcount	dockcount_AG	bikes_AG	take_AG	left_AG	total_take_bikes	total_left_bikes
BT-01	3rd Ave & Broad St	47.618418	-122.350964	18	15	9	[1]	[19]	352	338
BT-03	2nd Ave & Vine St	47.615829	-122.348564	16	16	11	[10]	0	313	222
BT-04	6th Ave & Blanchard St	47.616094	-122.341102	16	16	6	[1]	0	132	148
BT-05	2nd Ave & Blanchard St	47.613110	-122.344208	14	18	6	[2]	0	228	237
CBD-03	7th Ave & Union St	47.610731	-122.332447	20	22	8	0	0	220	279
CBD-04	Union St & 4th Ave	47.609221	-122.335596	18	14	6	0	0	0	0
CBD-05	1st Ave & Marion St	47.604058	-122.335800	20	23	8	0	0	246	374
CBD-06	2nd Ave & Spring St	47.605950	-122.335768	18	23	8	0	[5]	233	374
CBD-07	City Hall / 4th Ave & James St	47.603509	-122.330409	20	21	9	0	0	131	132
CBD-13	2nd Ave & Pine St	47.610185	-122.339641	18	23	8	0	[11]	394	555
CH-01	Summit Ave & E Denny Way	47.618633	-122.325249	16	20	10	0	0	190	111
CH-02	E Harrison St & Broadway Ave E	47.622063	-122.321251	20	16	8	[69]	0	317	158
CH-03	Summit Ave E & E Republican St	47.623367	-122.325279	16	19	9	[2]	0	163	89
CH-05	15th Ave E & E Thomas St	47.620712	-122.312805	16	19	10	[50]	0	273	69
CH-06	12th Ave & E Denny Way	47.618549	-122.317017	16	22	9	[4]	0	116	37
CH-07	E Pine St & 16th Ave	47.615330	-122.311752	18	16	9	[160]	0	401	98
CH-08	Cal Anderson Park / 11th Ave & Pine St	47.615486	-122.318245	26	16	11	[3]	0	292	161
CH-09	Harvard Ave & E Pine St	47.615517	-122.322083	16	17	11	[1]	0	199	138
CH-12	Bellevue Ave & E Pine St	47.615456	-122.326729	14	22	9	[2]	0	181	95
CH-15	12th Ave & E Mercer St	47.624142	-122.316811	16	19	10	[19]	0	235	88
DPD-01	9th Ave N & Mercer St	47.624298	-122.339617	18	21	6	0	[1]	183	262
DPD-03	Children's Hospital / Sandpoint Way NE & 40th ...	47.663509	-122.284119	24	21	11	0	0	72	75
EL-01	Fred Hutchinson Cancer Research Center / Fairv...	47.627643	-122.332576	16	19	8	0	[1]	153	167
EL-03	E Blaine St & Fairview Ave E	47.634831	-122.326634	18	22	7	0	0	198	267
EL-05	Eastlake Ave E & E Allison St	47.649090	-122.322983	18	17	8	0	0	133	149
FH-04	Seattle University / E Columbia St & 12th Ave	47.609239	-122.316651	20	23	10	0	0	100	66
ID-04	6th Ave S & S King St	47.598488	-122.326412	16	23	6	0	0	84	143
PS-04	Occidental Park / Occidental Ave S & S Washing...	47.600757	-122.332946	18	22	10	0	[32]	200	352
PS-05	King Street Station Plaza / 2nd Ave Extension ...	47.598994	-122.329684	18	18	6	0	[7]	137	252
SLU-01	REI / Yale Ave N & John St	47.619859	-122.330304	20	20	10	0	0	294	285
SLU-02	Dexter Ave N & Aloha St	47.627735	-122.342232	18	19	11	0	0	305	344
SLU-04	Republican St & Westlake Ave N	47.623165	-122.338203	18	22	7	0	[11]	226	381
SLU-07	PATH / 9th Ave & Westlake Ave	47.618320	-122.338913	18	23	9	0	[30]	292	470
SLU-15	Westlake Ave & 6th Ave	47.613628	-122.337341	20	14	7	[46]	[5]	443	365
SLU-16	Pine St & 9th Ave	47.613715	-122.331777	20	20	7	[6]	[24]	316	451
SLU-17	Lake Union Park / Valley St & Boren Ave N	47.626041	-122.335831	16	16	9	[3]	[10]	211	237
SLU-19	Key Arena / 1st Ave N & Harrison St	47.622277	-122.355230	16	22	10	[2]	0	286	318
SLU-20	Terry Ave & Stewart St	47.616260	-122.333815	20	17	9	0	0	0	0
SLU-21	Mercer St & 9th Ave N	47.624769	-122.339408	20	16	11	0	0	0	0
UD-01	Burke-Gilman Trail / NE Blakeley St & 24th Ave NE	47.666145	-122.301491	18	20	8	[1]	0	116	180
UD-02	NE 42nd St & University Way NE	47.658288	-122.313334	18	16	10	0	0	18	13
UD-04	12th Ave & NE Campus Pkwy	47.656395	-122.315620	16	15	10	0	[7]	154	180
UD-07	NE 47th St & 12th Ave NE	47.663143	-122.315086	16	16	8	0	0	157	99
UW-02	Burke Museum / E Stevens Way NE & Memorial Way NE	47.659756	-122.310402	14	13	8	[1]	0	147	80
UW-04	15th Ave NE & NE 40th St	47.655590	-122.311890	16	16	10	0	[6]	116	139
UW-06	UW Engineering Library / E Stevens Way NE & Je...	47.654613	-122.304863	14	18	11	[1]	0	107	106
UW-07	UW Intramural Activities Building	47.653713	-122.302162	14	15	6	0	0	74	95
UW-10	UW Magnuson Health Sciences Center Rotunda / C...	47.650725	-122.311188	16	20	8	0	0	69	93
WF-01	Pier 69 / Alaskan Way & Clay St	47.614315	-122.354093	24	23	7	[13]	[7]	523	551
WF-04	Seattle Aquarium / Alaskan Way S & Elliott Bay...	47.607702	-122.341650	18	20	6	0	[6]	189	273
CH-16	Broadway and E Denny Way	47.618640	-122.320777	18	22	7	0	0	0	0
SLU-22	Thomas St & 5th Ave N	47.620879	-122.347377	18	14	9	0	0	0	0
UW-11	NE Pacific St/UW Medical Center	47.649952	-122.306263	16	23	8	0	0	0	0

Figura 35: Resultats de cada estació per l'execució de 50 generacions

Els valors de dockcount_AG i bikes_AG són els valors dels gens del millor cromosoma, mentre que els valors take_AG i left_AG són els valors de la funció objectiu pel mes de Maig a cada estació per aquell cromosoma. Els valors totals de take i left són el nombre total de viatges que es van realitzar en aquell mes per a cada estació. Hi ha 7 estacions que no van tenir cap viatge, per tant, el valor de la funció objectiu sempre serà 0 independentment dels docks o bicis que hi posem. Tot i això, podem veure que el model obté bons resultats per a gairebé totes les estacions, tant sols hi ha 4 estacions que no s'han pogut iniciar o finalitzar més del 10% de viatges del mes. Destacar l'estació 'WF-01' que té més de 500 viatges iniciats i finalitzats al llarg del mes i tant sols 20 viatges no s'han pogut realitzar. En canvi, trobem interessant destacar l'estació 'CH-07' que té molts més viatges iniciats que de finalització que fa que tingui un valor de funció objectiu de take molt elevat de 160. Hi ha altres estacions que també tenen un comportament similar de des-balanceig entre viatges iniciats i finalitzats que fa que tinguin un valor alt en un dels dos valors de la funció objectiu. Veiem a continuació en quatre gràfics amb la localització de les estacions, els valors de les funcions objectius i el nombre total de viatges:

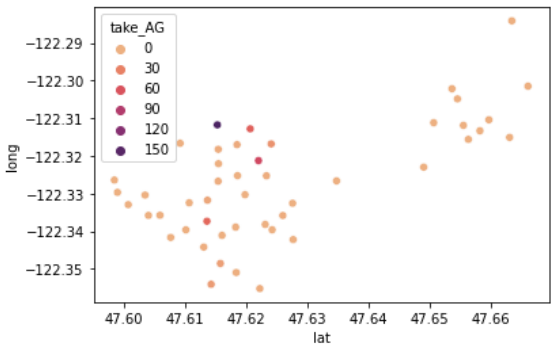

Figura 38: Funció objectiu de viatges iniciats per estació

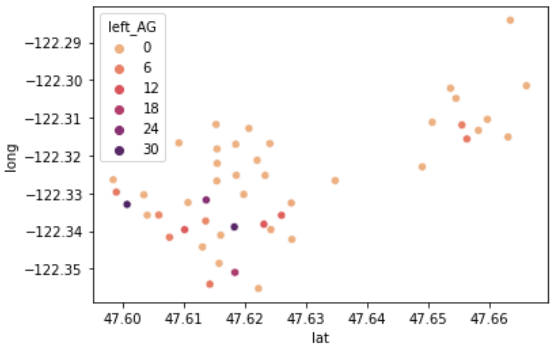

Figura 39: Funció objectiu de viatges finalitzats per estació

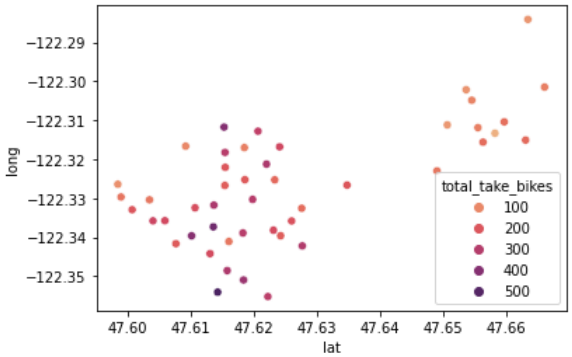

Figura 36: Total de viatges iniciats per estació

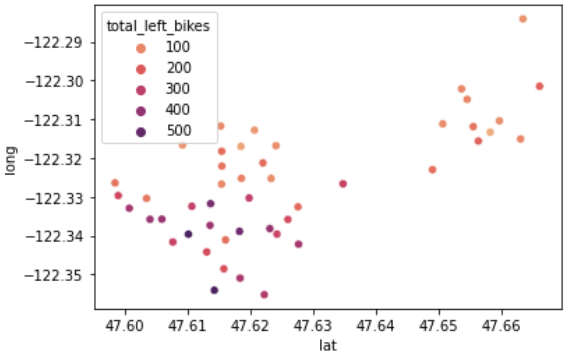

Figura 37: Total de viatges finalitzats per estació

D’aquesta forma podem identificar les estacions per les quals el model no ajusta bé i quines són les zones potencialment més probables de tenir viatges que no es poden realitzar. La zona de la dreta sembla que es balanceja bé ja que no hi ha un gran nombre d’usuaris, pel que no s’analitzarà amb més profunditat. La zona de l’esquerre, en canvi, és més complexa d’analitzar ja que hi ha un nombre més elevat d’usuaris que fan servir les diferents estacions.

Quan analitzem la zona de l’esquerre veiem que hi ha 3 estacions situades al nord que tenen problemes a l’hora d’agafar la bici però no per deixar-les. Una hipòtesi podria ser que es trobin situades en zones residencials i s’agafin per anar a altres punts de la ciutat amb més activitat, i a la tornada s’agafin altres transports. Podria ser també que hi hagués un fort pendent de baixada nord-sud i per aquest motiu els usuaris tant sols les agafin com a punt de partida. La zona central i sud són zones de molta activitat en el que el nombre de viatges que s’inicien i es finalitzen són similars. Veiem que només hi ha un punt on s’han trobat problemes per iniciar un trajecte, mentre que hi ha vuit punts on no s’han pogut finalitzar viatges. Aquests vuit punts, però, tenen al menys un punt proper on es podria finalitzar el viatge ja que són estacions que no han tingut viatges sense finalitzar. Per tant, veient aquests resultats sembla que pel mes de Maig tant sols hi ha una estació que sembla crítica degut al des-balanceig entre viatges iniciats i finalitzats, s’haurà de veure quan es faci el Test amb dades d’altres mesos si passa el mateix.

Tal i com s’ha detallat a l’apartat 4.4.1, s’ha executat el model sobre dades dels mesos de gener a maig de cara a veure si el model generalitza millor. Primer s’han executat 6 generacions del mes de maig, i, posteriorment els mesos de gener a maig amb 4 generacions per mes. Veiem l’evolució de la funció objectiu amb el canvi de mesos i amb el pas de les generacions:

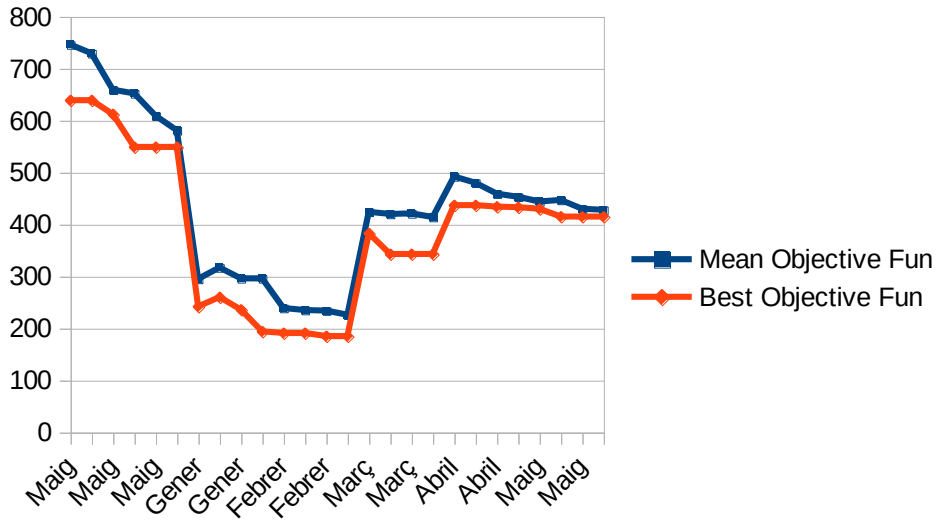

Figura 40: Execució de 26 generacions canviant el mes

Recordem que en aquest cas s'ha hagut de normalitzar el valor de la funció objectiu de cada mes dividint el resultat entre el total de desplaçaments d'aquells mes, per tant no es pot comparar el resultat obtingut amb els gràfics anteriors. Veiem que quan passem de maig a gener hi ha una baixada dràstica de la funció objectiu degut a un menor nombre de viatges. Aquesta baixada es manté el mes de febrer i puja de forma abrupta en el canvi al mes de març en què comencen a haver més viatges. Troba una solució prou òptima i torna a pujar amb el canvi al mes d'abril i la pujada de trajectes. És interessant que tot i que hi ha més trajectes al mes de maig, en aquets punt ja tenim una població força òptima amb millor resultats que la població que havíem aconseguit en les 6 primeres generacions. A l'apartat següent podrem validar si aquest resultat generalitza millor en mesos posteriors a maig del 2015.

Pintem també els gràfics amb el resultat de l'algorisme pel que fa als viatges no iniciats (take_AG) i els viatges no finalitzats (left_AG) i veim que els resultats són similars als obtinguts anteriorment però amb valors una mica més elevats:

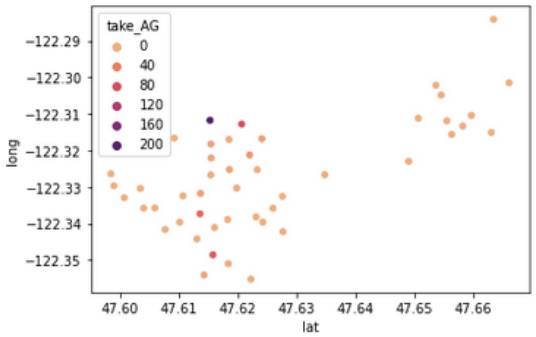

Figura 41: Funció objectiu de viatges iniciats per estació

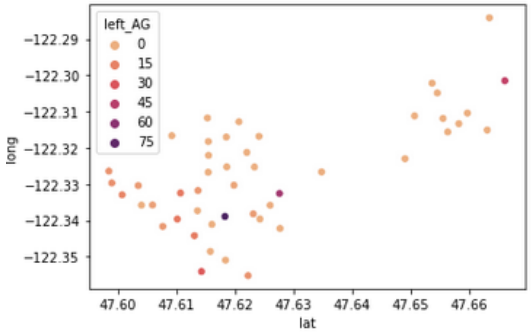

Figura 42: Funció objectiu de viatges finalitzats per estació

4.5.2 Test data

En aquest apartat s'utilitzarà els resultat obtinguts, és a dir la configuració dels diferents punts de bici, utilitzant dades de viatges d'altres mesos per veure si el resultat ha estat òptim. Es compararan 3 possibles solucions:

- Per defecte: Els docks tindran la mida del dataset original i el nombre de bicis que hi haurà a l'inici del dia serà la meitat del dock.
- Maig: Resultat de l'execució de 50 generacions pel mes de maig.
- Gener-Maig: Resultat de l'execució de 26 generacions, amb 6 generacions de maig i posteriorment 4 generacions per mes de gener a maig.

Aquestes tres possibles solucions s'han executat pels mesos des de juny del 2015 a agost del 2016, obtenint els següents resultats per la funció objectiu:

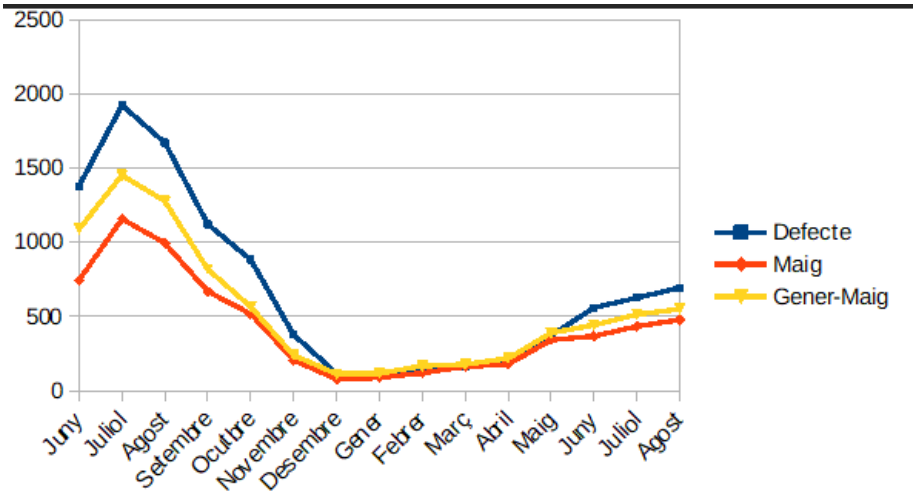

Figura 43: Evolució de les funcions objectius per mes

Si recordem el dataset de viatges, apartat 4.3.2, hi havia força menys viatges a l'any 2016 que explicaria el perquè d'un resultat de la funció objectiu més baix:

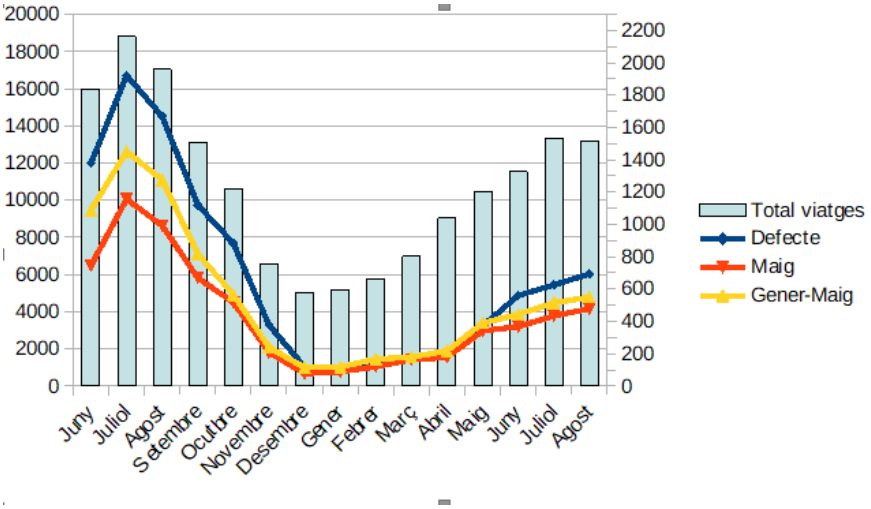

Figura 44: Evolució de les funcions objectius amb el total de viatges per mes

Tot i així, veiem que curiosament amb un número de viatges similar generalitzen millor pels mesos de maig a agost de 2016 que els mesos de setembre i octubre de 2015. Els mesos que hi ha pocs desplaçaments veiem que no importa quina solució apliquem, mentre que quan va augmentant el nombre de trajectes veiem que la millor solució és la que s'ha executat més generacions tot i haver-se executat per un únic mes. Veiem en el primer annex que les tres solucions tenen en molts casos valors similars per les diferents estacions, però aquestes petites diferències acumulades al llarg d'un mes fa que hi hagi un nombre bastant més elevat d'usuaris que insatisfets. Hi ha estacions en canvi que tenen valors molt diferents fent que en molts casos l'algoritme amb un òptim global sigui força pitjor

que les altres solucions. Per exemple, entre els mesos de juny i desembre de 2015 passem de tenir un total de 4368 viatges no iniciats o no finalitzats amb la solució més òptima, a tenir-ne 7472 amb la menys òptima (la solució per defecte), però en canvi l'estació 'CH-08' és bastant millor amb els valors per defecte que amb la solució a priori òptima. Això ens podria dir que han faltat generacions i més variabilitat a la població durant l'execució de l'algorisme.

Veiem el balanç de la funció objectiu per iniciar un viatge a les estacions per a cada solució pels mesos de juny a desembre de 2015:

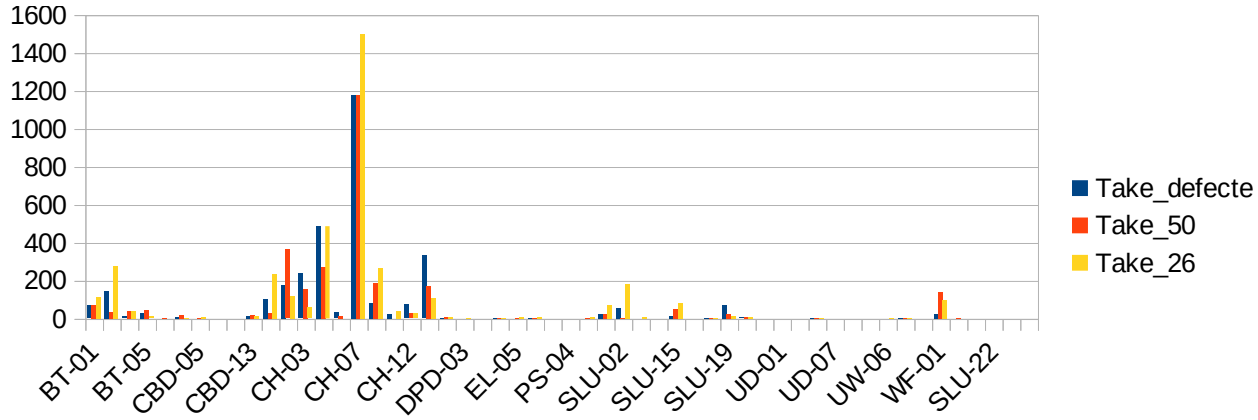

Figura 45: Balanç de la funció objectiu de viatges iniciats per estacions

Les estacions que tenen més viatges no iniciats són les mateixes però amb diferents volumetries. Destacar que en la execució de 26 generacions, tot i tenir una millor solució global, hi ha varies estacions que té pitjor resultat que la solució per defecte, entre elles l'estació 'CH-07' que ja havíem vista a l'apartat 4.5.1 on té molt pitjor resultat que la solució per defecte.

Analitzem també gràficament el balanç de finalització de viatges per estació pel període de juny a desembre de 2015 i veiem que és en aquest punt on es penalitza molt els valors per defecte:

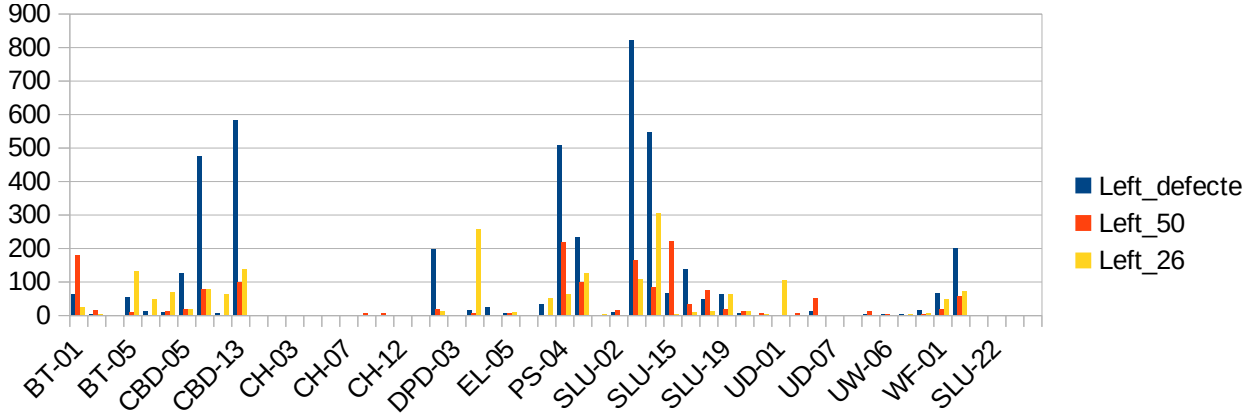

Figura 46:: Balanç de la funció objectiu de viatges finalitzats per estacions

5. Conclusions

En aquest treball s'ha realitzat una revisió i comprensió dels problemes d'optimització, els algoritmes evolutius i amb més profunditat els algoritmes genètics. S'ha realitzat un estudi de l'estat de l'art d'aquests tipus d'algorismes aplicats a resoldre problemes de mobilitat urbana, en concret a les xarxes de bike-sharing. S'ha vist que es tracta d'un problema molt complex de difícil solució ja que hi ha moltes estacions i molts usuaris, essent gairebé impossible trobar una configuració de la xarxa al principi del dia que faci que el flux d'entrades i sortides per estació no acabi en viatges fallits.

El plantejament del problema crec que ha estat adequat, així com el plantejament de la solució, però no ha estat del tot ben resolt a nivell de programació, ja que cada generació per una població no gaire gran (24 cromosomes) trigava en excés (~20 minuts). S'ha de tenir en compte que els cromosomes s'han hagut de crear amb dos al·lels per poder emmagatzemar tant la mida de l'estació com el número de bicis a l'iniciar el dia. Addicionalment, a cada generació s'havia de fer el balanç d'entrades i sortides per estació de forma seqüencial actualitzant l'estat de l'estació. El fet que els AG s'hagin d'executar generació per generació i que no es pugui passar a la següent fins que s'hagi avaluat l'actual, i que es facin servir per trobar solucions òptimes a problemes complexos fa que es tracti d'algoritmes costos computacionalment. En aquest cas especialment ja que dintre de cada generació s'ha hagut d'executar per a cada estació, com s'ha comentat anteriorment, el balanç d'entrades i sortides dintre d'un bucle.

Veient els resultats obtinguts i comparant amb els valors per defecte, apartat 4.5.2, crec que és un bon algoritme per a resoldre la problemàtica que es planteja ja que, amb una població no gaire gran i amb poques generacions ja s'han obtingut resultats globals molt millors que amb els valors per defecte. Els últims dos gràfics amb el valor de la funció objectiu per estació corroboren que si hi hagués hagut més variabilitat de cromosomes i més temps, s'hauria d'haver pogut trobar una solució que fos més òptima per a totes les estacions. S'ha trobat una solució molt òptima per moltes de les estacions amb restriccions pròpies de les dades (màxim de docks i de bicis), i s'han identificat quines són les estacions que necessiten que en algun moment del dia es recol·loquin bicis (buidar espais o omplir-los). S'ha plantejat una funció objectiu i unes restriccions molt senzilles que han funcionat força bé, però es podia haver afegit més complexitat penalitzant els llocs que són massa grans o els llocs o els que tenen bicis parades moltes hores. S'ha vist que el nombre de viatges totals té un pes molt important en la funció objectiu que s'ha plantejat, de fet si hi ha pocs viatges una configuració a l'atzar donaria resultats similars a una solució optimitzada; això segurament es podria haver afegit d'alguna forma a la funció objectiu. Per tant, podem concloure que plantejar una bona funció objectiu per un problema d'optimització no és una tasca trivial.

En línies generals s'han assolit tots els objectius que es van plantejar abans de realitzar el treball, aconseguint aplicar un algoritme genètic propi a un problema real amb bons resultats. Addicionalment, 'han assimilat correctament els conceptes que es volia i s'ha aconseguit tractar les dades de forma adequada al problema a resoldre. Tot i que no era un objectiu principal, tal i com s'ha mencionat anteriorment el codi no és tot l' òptim que es podria, es podria haver paral·lelitzat les execucions per a cada estació i no fer-les seqüencials per exemple, això hagués fet que s'hagués pogut executar l'algoritme en poblacions més grans i durant més generacions i veure realment tot el potencial dels AG. Tot i això, aquest fet m'ha permès veure l'alt cost computacional d'aquest tipus d'algoritmes, fet molt important en la resolució de problemes d'optimització on s'ha de trobar la millor solució possible en un temps que es consideri adient.

El plantejament inicial no va ser del tot adequat, on es va plantejar un treball molt genèric que abarcava massa, però va ser reconduït i enfocat de forma correcte. Un cop ben plantejat el treball, la planificació de tasques tampoc va ser del tot correcte i es va haver de moure algunes a entregues futures. Malgrat aquest fet, s'ha seguit la planificació de forma prou acurada fent que s'hagin assolit totes les tasques i tots els objectius planificats i s'ha sabut replanificar correctament.

Com a línies de treball futur estaria en primer lloc la re-factorització del codi per a poder fer execucions amb més cromosomes i més generacions en un temps adient. En segon lloc, un aspecte que s'ha trobat interessant mentre es desenvolupava el treball i que finalment no s'ha pogut realitzar és la agrupació d'estacions amb un algoritme de clustering tal i com s'ha vist a l'estat de l'art que realitzen altres autors. Això permetria obrir noves vies d'anàlisi o execució de l'algoritme adaptat a cada classe per exemple. També seria interessant treballar en la millora de la funció objectiu que permetes tenir en compte el nombre de viatges o que afegís la penalització per espais buits o bicis que no es mouen. Ha quedat pendent també com a línia futura veure com es comporta l'algoritme en diferents franges horàries[8]. Finalment, el següent pas seria el que realitzen la majoria d'articles de l'estat de l'art i afegir el fet que hi hagi una recol·locació de les bicis en algunes hores del dia. Aquest últim punt es podria combinar amb el punt de fer clustering i analitzar la recol·locació intra-clúster i inter-clústers [5].

6. Glossari

AG	Algoritme genètic
Al·lel	Mot biològic que aplicat als AG, i concretament al model desenvolupat, fa referència a una de les parts d'un gen
Bike sharing	Xarxa de bicis urbanes
Cromosoma	Mot biològic que aplicat als AG, possible solució que forma part d'una població
Crossover	Procés d'encreuament de dos cromosomes en un AG per obtenir-ne de nous
Data set	Conjunt de dades estructurades
Docks	Espais en els punts de bicis
Early Stop	Finalitzar abans d'hora un algoritme
Estocàstic	Procés no determinista on un estat no determina plenament el següent, és a dir que poden haver resultats diferents amb uns mateixos inputs.
Fitness	Com de bona és una solució
Gen	Mot biològic que aplicat als AG, i concretament al model desenvolupat, fa referència a una possible solució per una estació de bicis concreta.
Input	Entrada de dades
Output	Sortida de dades
Població	Conjunt de cromosomes

7. Bibliografia

[1] **Ali Askari. E, Bashiri. M**, (2017), A capacitated bike sharing location-allocation problem under demand uncertainty using sample average approximation: A greedy Genetic-Particle Swarm Optimization algorithm, Scientia Iranica, Volume 24, Issue 5, pp.2567-2580, doi: https://dx.doi.org/10.24200/sci.2017.4391

[2] **Ciencia de datos (autor: Joaquín Amat Rodrigo)**, (2019), Optimización con algoritmo genético y Nelder-Mead, consultat 17 d'octubre des de https://www.cienciadedatos.net/documentos/48_optimizacion_con_algoritmo_genetico

[3] **Dinu. S, Bordea. G**, (2011), A new genetic approach for transport network design and optimization , Bulletin of the polish Academy of Sciences, Technical Sciences, Volumen 59, Issue 3, doi: 10.2478/v10175-011-0032-z

[4] **Everitt. B**, (1987), Introduction to Optimization Methods and their Application in Statistics, consultat des de https://books.google.es/books?hl=en&lr=&id=Z04IBgAAQBAJ&oi=fnd&pg=PT6&dq=introduction+to+optimization&ots=RgIHfxvHln&sig=cs9jf9NsJY-oXswsG_to0e3s9Ms#v=onepage&q=introduction%20to%20optimization&f=false

[5] **Gurobi Optimization**, What is Mathematical Optimization?, consultat 10 d'octubre de 2021 des de https://www.gurobi.com/resources/mathematical-optimization-web-page/

[6] **Hannah. L**, (2014), Stochastic Optimization, consultat 15 d'octubre des de http://www.stat.columbia.edu/~liam/teaching/compstat-spr14/lauren-notes.pdf

[7] **Jian. N, Freund. D, Wiberg. H, Henderson. S**, (216), "Simulation optimization for a large-scale bike-sharing system," *Winter Simulation Conference (WSC)*, 2016, pp. 602-613, doi: 10.1109/WSC.2016.7822125.

[8] **Lahoorpoor. B, Faroqi, H, Sadeghi-Niaraki. A, Choi. S**, (2019), Spatial Cluster-Based Model for Static Rebalancing Bike Sharing Problem, Sustainability 2019, Volume 11, Issue 11, 3205, doi: https://doi.org/10.3390/su11113205

[9] **Martínez Ortega, C.C,** (2017). Introducción a la optimización robusta. (Trabajo Fin de Grado Inédito). Universidad de Sevilla, Sevilla, consultat 12 d'octubre de 2021 des de https://idus.us.es/handle/11441/63130

[10] **O'Mahony. E, Shmoys. D,** (2015), Data Analysis and Optimization for (Citi)Bike Sharing. *Proceedings of the AAAI Conference on Artificial Intelligence*, *29*(1). Consultat des de https://ojs.aaai.org/index.php/AAAI/article/view/9245

[11] **Romero. J. P**, (2013), Modelos de optimización para planificación y gestión opeartiva de sistemas de bicicleta pública (Tesis doctoral), Universidad de Cantabria, Cantabria

[12] **Sayarshad. H, Tavassoli. S, Zhao. F**, (2012), A multi-periodic optimization formulation for bike planning and bike utilization, Applied Mathematical Modelling, Volume 36, Issue 10, pp.4944-4951, doi: https://doi.org/10.1016/j.apm.2011.12.032

[13] **Stanford.edu**, Introduction to Mathematical Optimization, consultat 10 d'octubre de 2021 des de https://web.stanford.edu/group/sisl/k12/optimization/MO-unit1-pdfs/1.1optimization.pdf

[14] **Towards Data Science (autor: Devin Soni)**, (2018), Introduction to Evolutionary Algorithms, consultat 2 d'octubre des de https://towardsdatascience.com/introduction-to-evolutionary-algorithms-a8594b484ac

[15] **Ubalt.edu**, Modelos Deterministas: Optimización Lineal, consultat 15 d'octubre des de http://home.ubalt.edu/ntsbarsh/Business-stat/opre/SpanishD.htm

[16] **Walteros. J, Medaglia. A, Riaño. G**, (2013), Hybrid Algorithm for Route Design on Bus Rapid Transit Systems, Transportation Science, Articles in Advance, pp. 1-19, doi: http://dx.doi.org/10.1287/trsc.2013.0478

[17] **Wikipedia**, Mathematical optimization, consultat 10 d'octubre de 2021 des de https://en.wikipedia.org/wiki/Mathematical_optimization

8. Annexos

Annex1: taula resum amb els resultats dels cromosomes, al·lel 1 amb els docks de l'estació i al·lel 2 amb les bicis a l'inici del dia, per les tres solucions analitzades a l'apartat 4.5.2. Tenim la solució per defecte, la solució amb 50 generacions (maig) i la solució amb 26 generacions (gener-maig):

station_id	dockcount	Dock_defecte	Dock_50	Dock_26	Bikes_defecte	Bikes_50	Bikes_26
BT-01	18	18	15	20	9	9	8
BT-03	16	16	16	14	8	11	6
BT-04	16	16	16	19	8	6	6
BT-05	14	14	18	14	7	6	9
CBD-03	20	20	22	19	10	8	11
CBD-04	18	18	14	16	9	6	11
CBD-05	20	20	23	22	10	8	7
CBD-06	18	18	23	21	9	8	6
CBD-07	20	20	21	14	10	9	9
CBD-13	18	18	23	23	9	8	9
CH-01	16	16	20	22	8	10	6
CH-02	20	20	16	20	10	8	11
CH-03	16	16	19	15	8	9	11
CH-05	16	16	19	21	8	10	8
CH-06	16	16	22	13	8	9	11
CH-07	18	18	16	22	9	9	7
CH-08	26	26	16	18	13	11	10
CH-09	16	16	17	17	8	11	7
CH-12	14	14	22	23	7	9	9
CH-15	16	16	19	18	8	10	11
DPD-01	18	18	21	22	9	6	6
DPD-03	24	24	21	17	12	11	6
EL-01	16	16	19	14	8	8	11
EL-03	18	18	22	23	9	7	8
EL-05	18	18	17	14	9	8	6
FH-04	20	20	23	21	10	10	8
ID-04	16	16	23	17	8	6	10
PS-04	18	18	22	22	9	10	7
PS-05	18	18	18	22	9	6	11
SLU-01	20	20	20	16	10	10	7
SLU-02	18	18	19	21	9	11	7
SLU-04	18	18	22	22	9	7	6
SLU-07	18	18	23	22	9	9	11
SLU-15	20	20	14	23	10	7	6
SLU-16	20	20	20	22	10	7	6
SLU-17	16	16	16	20	8	9	9
SLU-19	16	16	22	19	8	10	11
SLU-20	20	20	17	16	10	9	8
SLU-21	20	20	16	16	10	11	10
UD-01	18	18	20	12	9	8	9
UD-02	18	18	16	19	9	10	6
UD-04	16	16	15	22	8	10	8
UD-07	16	16	16	22	8	8	8

UW-02	14	14	13	17	7	8	8
UW-04	16	16	16	21	8	10	6
UW-06	14	14	18	20	7	11	6
UW-07	14	14	15	15	7	6	8
UW-10	16	16	20	21	8	8	10
WF-01	24	24	23	21	12	7	8
WF-04	18	18	20	22	9	6	9
CH-16	18	18	22	15	9	7	9
SLU-22	18	18	14	23	9	9	9
UW-11	16	16	23	12	8	8	6
WF-03	18	18	20	18	9	7	10